PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

# INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G07F 7/10, 7/08, G06F 15/21

A1

(11) International Publication Number: WO 94/06103

(43) International Publication Date: 17 March 1994 (17.03.94)

(21) International Application Number: PCT/US93/08400

(22) International Filing Date: 7 September 1993 (07.09.93)

(30) Priority data:
07/941,971 8 September 1992 (08.09.92) US

(71) Applicant: HNC, INC. [US/US]; 5501 Oberlin Drive, San Diego, CA 92121 (US).

(72) Inventors: GOPINATHAN, Krishna, M. ; 9924 Kika Court, #2416, San Diego, CA 92129 (US). BIAFORE, Louis, S. ; 5075 La Cuenta Drive, San Diego, CA 92129 (US). FERGUSON, William, M. ; 4215 Feather Avenue, San Diego, CA 92117 (US). LAZARUS, Michael, A. ; 5030 Lotus Street, #57, San Diego, CA 92107 (US). PATHRIA, Anu, K. ; 5445 Locksley Avenue, Oakland, CA 94618 (US).

(74) Agents: RAUBVOGEL, Amir, H. et al.; Fenwick & West, 2 Palo Alto Square, Suite 500, Palo Alto, CA 94306 (US).

(81) Designated States: AT, AU, BB, BG, BR, BY, CA, CH, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

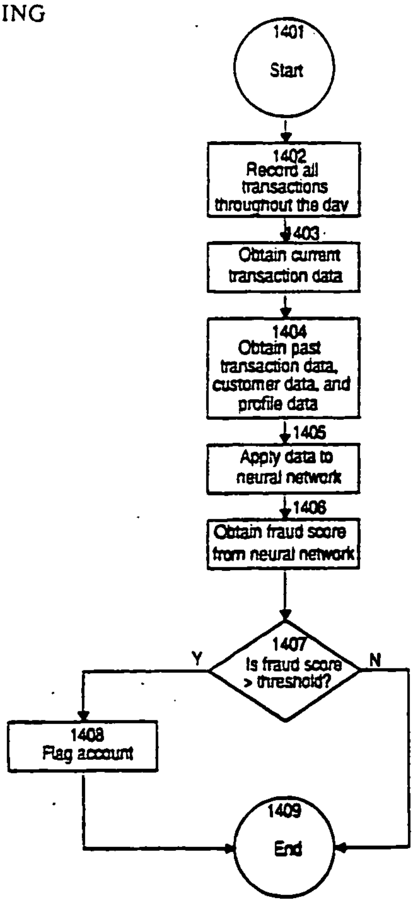
**Published**

*With international search report.*
*Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.*

## (54) Title: FRAUD DETECTION USING PREDICTIVE MODELING

### (57) Abstract

An automated system and method detects fraudulent transactions using a predictive model such as a neural network to evaluate indiviual customer accounts and identify potentially fraudulent transactions based on learned relationships among known variables. The system may also output reason codes indicating relative contributions of various variables to a particular result. The system periodically monitors its performance and redevelops the model when performance drops below a predetermined level.


1401
Start
1402
Record all transactions throughout the day
1403
Obtain current transaction data
1404
Obtain past transaction data, customer data, and profile data
1405
Apply data to neural network
1406
Obtain fraud score from neural network
1407
Is fraud score > threshold?
Y
N
1408
Flag account
1409
End

*FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

| | |
|---|---|
| AT | Austria |
| AU | Australia |
| BB | Barbados |
| BE | Belgium |
| BF | Burkina Faso |
| BG | Bulgaria |
| BJ | Benin |
| BR | Brazil |
| BY | Belarus |
| CA | Canada |
| CF | Central African Republic |
| CG | Congo |
| CH | Switzerland |
| CI | Côte d'Ivoire |
| CM | Cameroon |
| CN | China |
| CS | Czechoslovakia |
| CZ | Czech Republic |
| DE | Germany |
| DK | Denmark |
| ES | Spain |
| FI | Finland |
| FR | France |
| GA | Gabon |
| GB | United Kingdom |
| GN | Guinea |
| GR | Greece |
| HU | Hungary |
| IE | Ireland |
| IT | Italy |
| JP | Japan |
| KP | Democratic People's Republic of Korea |
| KR | Republic of Korea |
| KZ | Kazakhstan |
| LI | Liechtenstein |
| LK | Sri Lanka |
| LU | Luxembourg |
| LV | Latvia |
| MC | Monaco |
| MG | Madagascar |
| ML | Mali |
| MN | Mongolia |
| MR | Mauritania |
| MW | Malawi |
| NE | Niger |
| NL | Netherlands |
| NO | Norway |
| NZ | New Zealand |
| PL | Poland |
| PT | Portugal |
| RO | Romania |
| RU | Russian Federation |
| SD | Sudan |
| SE | Sweden |
| SI | Slovenia |
| SK | Slovak Republic |
| SN | Senegal |
| TD | Chad |
| TG | Togo |
| UA | Ukraine |
| US | United States of America |
| UZ | Uzbekistan |
| VN | Viet Nam |

# FRAUD DETECTION USING PREDICTIVE MODELING

## CROSS-REFERENCE TO RELATED APPLICATION

The subject matter of this application is related to the subject matter of pending U.S. application Serial No. 07/814,179, (attorney's docket number 726) for "Neural Network Having Expert System Functionality", by Curt A. Levey, filed December 30, 1991, the disclosure of which is incorporated herein by reference.

## BACKGROUND OF THE INVENTION

### 1. Field of the Invention

This invention relates generally to the detection of fraudulent use of customer accounts and account numbers, including for example credit card transactions. In particular, the invention relates to an automated fraud detection system and method that uses predictive modeling to perform pattern recognition and classification in order to isolate transactions having high probabilities of fraud.

### 2. Description of the Related Art

In the following discussion, the term "credit card" will be used for illustrative purposes; however, the techniques and principles discussed herein apply to other types of customer accounts, such as charge cards, bank automated teller machine cards and telephone calling cards.

Credit card issuers conventionally attempt to limit fraud losses by immediately closing a customer's account upon receiving a report that the card has been lost or stolen. Typically, the customer's credit information is then transferred to a new account and a new card is issued. This procedure is only effective in limiting fraudulent use of lost or stolen cards after the loss or theft has been reported to the issuer.

In many cases, however, fraudulent use occurs without the knowledge of the cardholder, and therefore no report is made to the issuer. This may occur if the customer is unaware that the card has been lost or stolen, or if other techniques are employed to perpetrate the fraud, such as: use of counterfeit cards; merchant fraud; application fraud; or interception of credit cards in the mail. In all these situations, the fraudulent use may not be detected until (and unless) the cardholder notices an unfamiliar transaction on his or her next monthly statement and contests the corresponding charge. The concomitant delay in detection of fraud may result in significant losses. User fraud, in which the user claims that a valid transaction is invalid, is also possible.

Issuers of credit cards have sought to limit fraud losses by attempting to detect fraudulent use before the cardholder has reported a lost or stolen card. One conventional technique is known as parameter analysis. A parameter analysis fraud detection scheme makes a decision using a small number of database fields combined in a simple Boolean condition. An example of such a condition is:

```
    if (number of transactions in 24 hours > X) and (more than Y dollars authorized)
then flag this account as high risk
```

Parameter analysis will provide the values of X and Y that satisfy either the required detection rate or the required false positive rate. In a hypothetical example, parameter values of X=400 and Y=1000 might capture 20% of the frauds with a false positive rate of 200:1, while X=6 and Y=2000 might capture 8% of the frauds with a false positive rate of 20:1.

The rules that parameter analysis provides are easily implemented in a database management system, as they are restricted to Boolean (e.g., and, or) combinations of conditions on single variables.

Parameter analysis derives rules by examining the single variables most able to distinguish fraudulent from non-fraudulent behavior. Since only single-variable threshold comparisons are used, complex interactions among variables are not captured. This is a limitation that could cause the system to discriminate poorly between fraudulent and valid account behavior, resulting in low capture rates and high false-positive rates.

Additionally, an effective fraud detection model generally requires more variables than conventional parameter analysis systems can handle. Furthermore, in order to capture new fraud schemes, parameter analysis systems must be redeveloped often, and automated redevelopment is difficult to implement.

It is desirable, therefore, to have an automated system that uses available information regarding cardholders, merchants, and transactions to screen transactions and isolate those which are likely to be fraudulent, and which captures a relatively high proportion of frauds while maintaining a relatively low false-positive rate. Preferably, such a system should be able to handle a large number of interdependent variables, and should have capability for redevelopment of the underlying system model as new patterns of fraudulent behavior emerge.

## SUMMARY OF THE INVENTION

In accordance with the present invention, there is provided an automated system and method for detecting fraudulent transactions, which uses a predictive model such as a neural network to evaluate individual customer accounts and identify potentially fraudulent transactions based on learned relationships among known variables. These relationships enable the system to estimate a probability of fraud for each transaction. This probability may then be provided as output to a human decision-maker involved in processing the transaction, or the issuer may be signaled when the probability exceeds a predetermined amount. The system may also output reason codes that reveal the relative contributions of various factors to a particular result. Finally, the system periodically monitors its performance, and redevelops the model when performance drops below a predetermined level.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of an implementation of the present invention.

Figure 2 is a sample system monitor screen which forms part of a typical output interface for the present invention.

Figure 3 is a sample account selection screen which forms part of a typical output interface for the present invention.

Figure 4 is a sample transaction analysis screen which forms part of a typical output interface for the present invention.

Figure 5 is a sample customer information screen which forms part of a typical output interface for the present invention.

Figure 6 is a sample analyst response screen which forms part of a typical output interface for the present invention.

Figure 7 is a flowchart illustrating the major functions and operation of the present invention.

Figure 8 is a block diagram showing the overall functional architecture of the present invention.

Figure 9 is a diagram of a single processing element within a neural network.

Figure 10 is a diagram illustrating hidden processing elements in a neural network.

Figure 11 is a flowchart of the pre-processing method of the present invention.

Figure 12 is a flowchart of the method of creating a profile record of the present invention.

Figure 13 is a flowchart of the method of updating a profile record of the present invention.

Figure 14 is a flowchart showing operation of a batch transaction processing system according to the present invention.

Figure 15 is a flowchart showing operation of a semi-real-time transaction processing system according to the present invention.

Figure 16 is a flowchart showing operation of a real-time processing system according to the present invention.

Figure 17 is a flowchart showing the overall operation of the transaction processing component of the present invention.

Figure 18 is a flowchart showing the operation of module CSCORE of the present invention.

Figure 19 is a flowchart showing the operation of DeployNet of the present invention.

Figure 20 is a flowchart showing cascaded operation of the present invention.

Figure 21 is a portion of a typical CFG model definition file.

## Description of the Preferred Embodiment

The Figures depict preferred embodiments of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

Referring now to Figure 1, there is shown a block diagram of a typical implementation of a system 100 in accordance with the present invention. Transaction information is applied to system 100 via data network 105, which is connected to a conventional financial data facility 106 collecting transaction information from conventional sources such as human-operated credit-card authorization terminals and automated teller machines (not shown). CPU 101 runs software program instructions, stored in program storage 107, which direct CPU 101 to perform the various functions of the system. In the preferred embodiment, the software program is written in the ANSI C language, which may be run on a variety of conventional hardware platforms. In accordance with the software program instructions, CPU 101 stores the data obtained from data network 105 in data storage 103, and uses RAM 102 in a conventional manner as a workspace. CPU 101, data storage 103, and program storage 107 operate together to provide a neural network model 108 for predicting fraud. After neural network 108 processes the information, as described below, to obtain an indication of the likelihood of fraud, a signal indicative of that likelihood is sent from CPU 101 to output device 104.

In the preferred embodiment, CPU 101 is a Model 3090 IBM mainframe computer, RAM 102 and data storage 103 are conventional RAM, ROM and disk storage devices for the

Model 3090 CPU, and output device 104 is a conventional means for either printing results based on the signals generated by neural network 108, or displaying the results on a video screen using a window-based interface system, or sending the results to a database for later access, or sending a signal dependent on the results to an authorization system (not shown) for further processing.

Referring now also to Figures 2 through 6, there are shown sample screens from a conventional window-based interface system (not shown) which forms part of output device 104. Figure 2 shows system monitor 201 that allows a fraud analyst or system supervisor to review system performance. System monitor 201 shows a cutoff score 202 above which accounts will be flagged, the number of accounts with scores above the cutoff 203, and the fraud score 204 and account number 205 for a particular account.

Figure 3 shows account selection screen 301 that includes a scrolling window 302 allowing the analyst to select high-risk transactions for review, and a set of buttons 303 allowing the analyst to select further operations in connection with the selected transactions.

Figure 4 shows transaction analysis screen 401 that allows the fraud analyst to examine each high-risk transaction and determine appropriate fraud control actions. It includes account information 402, fraud score 403, explanations derived from reason codes 404 that indicate the reasons for fraud score 403, and two scrolling windows 405 and 406 that show transaction information for the current day and the past seven days 405, and for the past six months 406.

Figure 5 shows customer information screen 501 that allows the analyst to access customer information, including account number 502, customer names 503, best time to call 504, phone numbers 505, and address 506. It also provides access to further functions via on-screen buttons 507.

Figure 6 shows analyst response screen 601 that allows the analyst to log actions taken to control fraud. It includes a series of check boxes 602 for logging information, a comment box 603, and on-screen buttons 604 allowing access to other functions.

Referring now also to Figure 7, there is shown an overall flowchart illustrating the major functions and operation of the system 100. First neural network model 108 is trained 701 using data describing past transactions from data network 105. Then data describing the network model are stored 702. Once the model description is stored, system 100 is able to process current transactions. System 100 obtains data for a current transaction 703, and applies the current transaction data to the stored network model 704. The model 108 determines a fraud score and reason codes (described below), which are output 705 to the user, or to a database, or to another system via output device 104.

Referring now to Figure 8, the overall functional architecture of system 100 is shown. System 100 is broken down into two major components: model development component 801 and transaction processing component 802. Model development component 801 uses past data

804 to build neural network 108 containing information representing learned relationships among a number of variables. Together, the learned relationships form a model of the behavior of the variables. Although a neural network is used in the preferred embodiment, any type of predictive modeling technique may be used. For purposes of illustration, the invention is described here in terms of a neural network.

Transaction processing component 802 performs three functions: 1) it determines the likelihood of fraud for each transaction by feeding data from various sources 805, 806 into neural network 108, obtaining results, and outputting them 807; 2) when applicable, it creates a record in a profile database 806 summarizing past transactional patterns of the customer; and 3) when applicable, it updates the appropriate record in profile database 806.

Each of the two components of the system will be described in turn.

Model Development Component 801

Neural Networks: Neural networks employ a technique of "learning" relationships through repeated exposure to data and adjustment of internal weights. They allow rapid model development and automated data analysis. Essentially, such networks represent a statistical modeling technique that is capable of building models from data containing both linear and non-linear relationships. While similar in concept to regression analysis, neural networks are able to capture nonlinearity and interactions among independent variables without pre-specification. In other words, while traditional regression analysis requires that nonlinearities and interactions be detected and specified manually, neural networks perform these tasks automatically. For a more detailed description of neural networks, see D.E. Rumelhart et al, "Learning Representations by Back-Propagating Errors", Nature v. 323, pp. 533-36 (1986), and R. Hecht-Nielsen, "Theory of the Backpropagation Neural Network", in Neural Networks for Perception, pp. 65-93 (1992), the teachings of which are incorporated herein by reference.

Neural networks comprise a number of interconnected neuron-like processing elements that send data to each other along connections. The strengths of the connections among the processing elements are represented by weights. Referring now to Figure 9, there is shown a diagram of a single processing element 901. The processing element receives inputs $X_1$, $X_2$, ... $X_n$, either from other processing elements or directly from inputs to the system. It multiplies each of its inputs by a corresponding weight $w_1$, $w_2$, ... $w_n$ and adds the results together to form a weighted sum 902. It then applies a transfer function 903 (which is typically non-linear) to the weighted sum, to obtain a value Z known as the state of the element. The state Z is then either passed on to another element along a weighted connection, or provided as an output signal. Collectively, states are used to represent information in the short term, while weights represent long-term information or learning.

Processing elements in a neural network can be grouped into three categories: input processing elements (those which receive input data values); output processing elements (those which produce output values); and hidden processing elements (all others). The purpose of hidden processing elements is to allow the neural network to build intermediate representations that combine input data in ways that help the model learn the desired mapping with greater accuracy. Referring now to Figure 10, there is shown a diagram illustrating the concept of hidden processing elements. Inputs 1001 are supplied to a layer of input processing elements 1002. The outputs of the input elements are passed to a layer of hidden elements 1003. Typically there are several such layers of hidden elements. Eventually, hidden elements pass outputs to a layer of output elements 1004, and the output elements produce output values 1005.

Neural networks learn from examples by modifying their weights. The "training" process, the general techniques of which are well known in the art, involves the following steps:

1) Repeatedly presenting examples of a particular input/output task to the neural network model;

2) Comparing the model output and desired output to measure error; and

3) Modifying model weights to reduce the error.

This set of steps is repeated until further iteration fails to decrease the error. Then, the network is said to be "trained." Once training is completed, the network can predict outcomes for new data inputs.

## Fraud-Related Variables

In the present invention, data used to train the model are drawn from various database files containing historical data on individual transactions, merchants, and customers. These data are preferably pre-processed before being fed into the neural network, resulting in the creation of a set of fraud-related variables that have been empirically determined to form more effective predictors of fraud than the original historical data.

Referring now to Figure 11, there is shown a flowchart of the pre-processing method of the present invention. Individual elements of the flowchart are indicated by designations which correspond to module names.

Data used for pre-processing is taken from three databases which contain past data: 1) past transaction database 1101 (also called an "authorization database") containing two years' worth of past transaction data, which may be implemented in the same data base as past data 804; 2) customer database 1103 containing customer data; and 3) fraud database 1102 which indicates which accounts had fraudulent activity and when the fraudulent activity occurred.

Module readauth.sas 1104 reads transaction data from past transaction database 1101. Module matchauth.sas 1105 samples this transaction data to obtain a new transaction data set containing all of the fraud accounts and a randomly-selected subset of the non-fraud accounts. In creating the new transaction data set, module matchauth.sas 1105 uses information from fraud database 1102 to determine which accounts have fraud and which do not. For effective network training, it has been found preferable to obtain approximately ten non-fraud accounts for every fraud account.

Module readex.sas 1106 reads customer data from customer database 1103. Module matchex.sas 1107 samples this customer data to obtain a new customer data set containing all of the fraud accounts and the same subset of non-fraud accounts as was obtained by module matchauth.sas. In creating the new customer data set, module matchex.sas 1107 uses information from fraud database 1102 to determine which accounts have fraud and which do not.

Module mxmerge.sas 1108 merges all of the data sets obtained by modules matchauth.sas 1105 and matchex.sas 1107. Module genau.sas 1109 subdivides the merged data set into subsets of monthly data.

Module gensamp.sas 1112 samples the data set created by module mxmerge.sas 1108 and subdivided by genau.sas 1109, and creates a new data set called sample.ssd where each record represents a particular account on a particular day with transaction activity. Module gensamp.sas 1112 determines which records are fraudulent using information from fraud database 1102. Module gensamp.sas 1112 provides a subset of authorization days, as follows: From the database of all transactions, a set of active account-days is created by removing multiple transactions for the same customer on the same day. In the set of active account-days, each account day is assigned a "draft number" from 0 to 1. This draft number is assigned as follows: If the account-day is non-fraudulent, then the draft number is set to a random number between 0 and 1. If the account-day is fraudulent and it lies on the first or second day of fraud, then the draft number is set to 0. Otherwise, it is set to 1. Then, the 25,000 account-days with the smallest draft numbers are selected for inclusion in sample.ssd. Thus, all fraudulent account-days (up to 25,000) plus a sample of non-fraudulent account-days are included in sample.ssd.

Module roll15.sas 1113 generates a 15-day rolling window of data. This data has multiple records for each account-day listed in sample.ssd. The current day and 14 preceding days are listed for each sample account.

Module roll15to7.sas 1117 takes the roll15 data set and filters out days eight to 15 to produce roll7, a 7-day rolling window data set 1119. Days eight to 15 are ignored. Module genrolv.sas 1118 generates input variables for a rolling window of the previous 15 days of transactions. It processes a data set with multiple and variable numbers of records per account and produces a data set with one record per account. The result is called rollv.ssd.

Module roll15to1.sas 1114 takes the roll15 data set and filters out days except the current day to produce roll1. Module gencurv.sas 1115 uses roll1 to generate current day variables 1116 describing transactions occurring during the current day.

Module genprof.sas generates profile variables which form the profile records 1111.

Module merge.sas 1120 combines the profile records 1111, 1-day variables 1116, and 7-day variables 1119 and generates new fraud-related variables, as listed below, from the combination. It also merges rollv.ssd with the sample-filtered profile data sets to produce a single data set with both profile and rolling window variables. The result is called the mod1n2 data set 1121 (also called the "training set"), which contains the fraud-related variables needed to train the network. Scaler module 1122 scales the variables such that the mean value for each variable in the scaled training set is 0.0 and the standard deviation is 1.0, to create scaled mod1n2 data set 1123.

Many fraud-related variables may be generated using variations of the pre-processing method described above. Fraud-related variables used in the preferred embodiment include:

- Customer usage pattern profiles representing time-of-day and day-of-week profiles;
- Expiration date for the credit card;
- Dollar amount spent in each SIC (Standard Industrial Classification) merchant group category during the current day;
- Percentage of dollars spent by a customer in each SIC merchant group category during the current day;
- Number of transactions in each SIC merchant group category during the current day;
- Percentage of number of transactions in each SIC merchant group category during the current day;
- Categorization of SIC merchant group categories by fraud rate (high, medium, or low risk);
- Categorization of SIC merchant group categories by customer types (groups of customers that most frequently use certain SIC categories);
- Categorization of geographic regions by fraud rate (high, medium, or low risk);
- Categorization of geographic regions by customer types;
- Mean number of days between transactions;
- Variance of number of days between transactions;
- Mean time between transactions in one day;
- Variance of time between transactions in one day;
- Number of multiple transaction declines at same merchant;
- Number of out-of-state transactions;
- Mean number of transaction declines;
- Year-to-date high balance;
- Transaction amount;
- Transaction date and time;
- Transaction type.

Additional fraud-related variables which may also be considered are listed below:

Current Day Cardholder Fraud Related Variables

bweekend - current day boolean indicating current datetime considered weekend
cavapvdl - current day mean dollar amount for an approval
cavapvdl - current day mean dollar amount for an approval
cavaudl - current day mean dollars per auth across day
ccoscdom - current day cosine of the day of month i.e. cos(day ((datepart(cst_dt) * &TWOPI)/30));
ccoscdow - current day cosine of the day of week i.e. cos(weekday((datepart(cst_dt) * &TWOPI)/7));
ccoscmoy - current day cosine of the month of year i.e. cos(month ((datepart(cst_dt) * &TWOPI)/12));
cdom - current day day of month
cdow - current day day of week
chdzip - current cardholder zip
chibal - current day high balance
chidcapv - current day highest dollar amt on a single cash approve
chidcdec - current day highest dollar amt on a single cash decline
chidmapv - current day highest dollar amt on a single merch approve
chidmdec - current day highest dollar amt on a single merch decline
chidsapv - current day highest dollar amount on a single approve
chidsau - current day highest dollar amount on a single auth
chidsdec - current day highest dollar amount on a single decline
cmoy - current day month of year
cratdcau - current day ratio of declines to auths
csincdom - current day sine of the day of month i.e. sin(day ((datepart(cst_dt) * &TWOPI)/30));
csincdow - current day sine of the day of week i.e. sin(weekday((datepart(cst_dt) * &TWOPI)/7));
csincmoy - current day sine of the month of year i.e. sin(month ((datepart(cst_dt) * &TWOPI)/12));
cst_dt - current day cst datetime derived from zip code and CST auth time
ctdapv - current day total dollars of approvals
ctdau - current day total dollars of auths
ctdcsapv - current day total dollars of cash advance approvals
ctdcsdec - current day total dollars of cash advance declines

ctddec - current day total dollars of declines
ctdmrapv - current day total dollars of merchandise approvals
ctdmrdec - current day total dollars of merchandise declines
ctnapv - current day total number of approves
ctnau - current day total number of auths
ctnau10d - current day number of auths in day<=$10
ctnaudy - current day total number of auths in a day
ctncsapv - current day total number of cash advance approvals
ctncsapv - current day total number of cash approves
ctncsdec - current day total number of cash advance declines
ctndec - current day total number of declines
ctnmrapv - current day total number of merchandise approvals
ctnmrdec - current day total number of merchandise declines
ctnsdapv - current day total number of approvals on the same day of week as current day
ctnwdaft - current day total number of weekday afternoon approvals
ctnwdapv - current day total number of weekday approvals
ctnwdeve - current day total number of weekday evening approvals
ctnwdmor - current day total number of weekday morning approvals
ctnwdnit - current day total number of weekday night approvals
ctnweaft - current day total number of weekend afternoon approvals
ctnweapv current day total number of weekend approvals
ctnweeve current day total number of weekend evening approvals
ctnwemor current day total number of weekend morning approvals
ctnwenit current day total number of weekend night approvals
currbal current day current balance
cvrandl current day variance of dollars per auth across day
czrate1 current day zip risk group 1 'Zip very high fraud rate'
czrate2 current day zip risk group 2 'Zip high fraud rate'
czrate3 current day zip risk group 3 'Zip medium high fraud rate'
czrate4 current day zip risk group 4 'Zip medium fraud rate'
czrate5 current day zip risk group 5 'Zip medium low fraud rate'
czrate6 current day zip risk group 6 'Zip low fraud rate'
czrate7 current day zip risk group 7 'Zip very low fraud rate'
czrate8 current day zip risk group 8 'Zip unknown fraud rate'
ctdsfa01 current day total dollars of transactions in SIC factor group 01
ctdsfa02 current day total dollars of transactions in SIC factor group 02
ctdsfa03 current day total dollars of transactions in SIC factor group 03

| | |
|---|---|
| ctdsfa04 | current day total dollars of transactions in SIC factor group 04 |
| ctdsfa05 | current day total dollars of transactions in SIC factor group 05 |
| ctdsfa06 | current day total dollars of transactions in SIC factor group 06 |
| ctdsfa07 | current day total dollars of transactions in SIC factor group 07 |
| ctdsfa08 | current day total dollars of transactions in SIC factor group 08 |
| ctdsfa09 | current day total dollars of transactions in SIC factor group 09 |
| ctdsfa10 | current day total dollars of transactions in SIC factor group 10 |
| ctdsfa11 | current day total dollars of transactions in SIC factor group 11 |
| ctdsra01 | current day total dollars of transactions in SIC fraud rate group 01 |
| ctdsra02 | current day total dollars of transactions in SIC fraud rate group 02 |
| ctdsra03 | current day total dollars of transactions in SIC fraud rate group 03 |
| ctdsra04 | current day total dollars of transactions in SIC fraud rate group 04 |
| ctdsra05 | current day total dollars of transactions in SIC fraud rate group 05 |
| ctdsra06 | current day total dollars of transactions in SIC fraud rate group 06 |
| ctdsra07 | current day total dollars of transactions in SIC fraud rate group 07 |
| ctdsva01 | current day total dollars in SIC VISA group 01 |
| ctdsva02 | current day total dollars in SIC VISA group 02 |
| ctdsva03 | current day total dollars in SIC VISA group 03 |
| ctdsva04 | current day total dollars in SIC VISA group 04 |
| ctdsva05 | current day total dollars in SIC VISA group 05 |
| ctdsva06 | current day total dollars in SIC VISA group 06 |
| ctdsva07 | current day total dollars in SIC VISA group 07 |
| ctdsva08 | current day total dollars in SIC VISA group 08 |
| ctdsva09 | current day total dollars in SIC VISA group 09 |
| ctdsva10 | current day total dollars in SIC VISA group 10 |
| ctdsva11 | current day total dollars in SIC VISA group 11 |
| ctnsfa01 | current day total number of transactions in SIC factor group 01 |
| ctnsfa02 | current day total number of transactions in SIC factor group 02 |
| ctnsfa03 | current day total number of transactions in SIC factor group 03 |
| ctnsfa04 | current day total number of transactions in SIC factor group 04 |
| ctnsfa05 | current day total number of transactions in SIC factor group 05 |
| ctnsfa06 | current day total number of transactions in SIC factor group 06 |
| ctnsfa07 | current day total number of transactions in SIC factor group 07 |
| ctnsfa08 | current day total number of transactions in SIC factor group 08 |
| ctnsfa09 | current day total number of transactions in SIC factor group 09 |
| ctnsfa10 | current day total number of transactions in SIC factor group 10 |
| ctnsfa11 | current day total number of transactions in SIC factor group 11 |

| | |
|---|---|
| ctnsra01 | current day total number of transactons in SIC fraud rate group 01 |
| ctnsra02 | current day total number of transactons in SIC fraud rate group 02 |
| ctnsra03 | current day total number of transactons in SIC fraud rate group 03 |
| ctnsra04 | current day total number of transactons in SIC fraud rate group 04 |
| ctnsra05 | current day total number of transactons in SIC fraud rate group 05 |
| ctnsra06 | current day total number of transactons in SIC fraud rate group 06 |
| ctnsra07 | current day total number of transactons in SIC fraud rate group 07 |
| ctnsva01 | current day total number in SIC VISA group 01 |
| ctnsva02 | current day total number of SIC VISA group 02 |
| ctnsva03 | current day total number of SIC VISA group 03 |
| ctnsva04 | current day total number of SIC VISA group 04 |
| ctnsva05 | current day total number of SIC VISA group 05 |
| ctnsva06 | current day total number of SIC VISA group 06 |
| ctnsva07 | current day total number of SIC VISA group 07 |
| ctnsva08 | current day total number of SIC VISA group 08 |
| ctnsva09 | current day total number of SIC VISA group 09 |
| ctnsva10 | current day total number of SIC VISA group 10 |
| ctnsva11 | current day total number of SIC VISA group 11 |

7 Day Cardholder Fraud Related Variables

| | |
|---|---|
| raudymdy | 7 day ratio of auth days over number of days in the window |
| ravapvdl | 7 day mean dollar amount for an approval |
| ravaudl | 7 day mean dollars per auth across window |
| rddapv | 7 day mean dollars per day of approvals |
| rddapv2 | 7 day mean dollars per day of approvals on days with auths |
| rddau | 7 day mean dollars per day of auths on days with auths |
| rddauall | 7 day mean dollars per day of auths on all days in window |
| rddcsapv | 7 day mean dollars per day of cash approvals |
| rddcsdec | 7 day mean dolalrs per day of cash declines |
| rdddec | 7 day mean dollars per day of declines |
| rdddec2 | 7 day mean dollars per day of declines on days with auths |
| rddmrapv | 7 day mean dollars per day of merchandise approvals |
| rddmrdec | 7 day mean dollars per day of merchandise declines |
| rdnapv | 7 day mean number per day of approvals |
| rdnau | 7 day mean number per day of auths on days with auths |
| rdnauall | 7 day mean number per day of auths on all days in window |

| | |
|---|---|
| rdncsapv | 7 day mean number per day of cash approvals |
| rdncsdec | 7 day mean number per day of cash declines |
| rdndec | 7 day mean number per day of declines |
| rdnmrapv | 7 day mean number per day of merchandise approvals |
| rdnmrdec | 7 day mean number per day of merchandise declines |
| rdnsdap2 | 7 day mean number per day of approvals on same day of week calculated only for those days which had approvals |
| rdnsdapv | 7 day mean number per day of approvals on same day of week as current day |
| rdnwdaft | 7 day mean number per day of weekday afternoon approvals |
| rdnwdapv | 7 day mean number per day of weekday approvals |
| rdnwdeve | 7 day mean number per day of weekday evening approvals |
| rdnwdmor | 7 day mean number per day of weekday morning approvals |
| rdnwdnit | 7 day mean number per day of weekday night approvals |
| rdnweaft | 7 day mean number per day of weekend afternoon approvals |
| rdnweapv | 7 day mean number per day of weekend approvals |
| rdnweeve | 7 day mean number per day of weekend evening approvals |
| rdnwemor | 7 day mean number per day of weekend morning approvals |
| rdnwenit | 7 day mean number per day of weekend night approvals |
| rhibal | 7 day highest window balance |
| rhidcapv | 7 day highest dollar amt on a single cash approve |
| rhidcdec | 7 day highest dollar amt on a single cash decline |
| rhidmapv | 7 day highest dollar amt on a single merch approve |
| rhidmdec | 7 day highest dollar amt on a single merch decline |
| rhidsapv | 7 day highest dollar amount on a single approve |
| rhidsau | 7 day highest dollar amount on a single auth |
| rhidsdec | 7 day highest dollar amount on a single decline |
| rhidtapv | 7 day highest total dollar amount for an approve in a single day |
| rhidtau | 7 day highest total dollar amount for any auth in a single day |
| rhidtdec | 7 day highest total dollar amount for a decline in a single day |
| rhinapv | 7 day highest number of approves in a single day |
| rhinau | 7 day highest number of auths in a single day |
| rhindec | 7 day highest number of declines in a single day |
| rnaudy | 7 day number of days in window with any auths |
| rnausd | 7 day number of same day of week with any auths |
| rnauwd | 7 day number of weekday days in window with any auths |
| rnauwe | 7 day number of weekend days in window with any auths |
| rncsandy | 7 day number of days in window with cash auths |

| | |
|---|---|
| rnmraudy | 7 day number of days in window with merchant auths |
| rtdapv | 7 day total dollars of approvals |
| rtdau | 7 day total dollars of auths |
| rtdcsapv | 7 day total dollars of cash advance approvals |
| rtdcsdec | 7 day total dollars of cash advance declines |
| rtddec | 7 day total dollars of declines |
| rtdmrapv | 7 day total dollars of merchandise approvals |
| rtdmrdec | 7 day total dollars of merchandise declines |
| rtnapv | 7 day total number of approvals |
| rtnapvdy | 7 day total number of approves in a day |
| rtnau | 7 day total number of auths |
| rtnau10d | 7 day number of auths in window <=$10 |
| rtncsapv | 7 day total number of cash advance approvals |
| rtncsdec | 7 day total number of cash advance declines |
| rtndec | 7 day total number of declines |
| rtnmrapv | 7 day total number of merchandise approvals |
| rtnmrdec | 7 day total number of merchandise declines |
| rtnsdapv | 7 day total number of approvals on the same day of week as current day |
| rtnwdaft | 7 day total number of weekday afternoon approvals |
| rtnwdapv | 7 day total number of weekday approvals |
| rtnwdeve | 7 day total number of weekday evening approvals |
| rtnwdmor | 7 day total number of weekday morning approvals |
| rtnwdnit | 7 day total number of weekday night approvals |
| rtnweaft | 7 day total number of weekend afternoon approvals |
| rtnweapv | 7 day total number of weekend approvals |
| rtnweeve | 7 day total number of weekend evening approvals |
| rtnwemor | 7 day total number of weekend morning approvals |
| rtnwenit | 7 day total number of weekend night approvals |
| rvraudl | 7 day variance of dollars per auth across window |

Profile Cardholder Fraud Related Variables

| | |
|---|---|
| paudymdy | - profile ratio of auth days over number of days in the month |
| pavapvdl | - profile mean dollar amount for an approval |
| pavaudl | - profile mean dollars per auth across month |
| pchdzip | - profile the last zip of the cardholder |
| pdbm | - profile value of 'date became member' at time of last profile update |

| | |
|---|---|
| pddapv | - profile daily mean dollars of approvals |
| pddapv2 | - profile daily mean dollars of approvals on days with auths |
| pddau | - profile daily mean dollars of auths on days with auths |
| pddau30 | - profile daily mean dollars of auths on all days in month |
| pddcsapv | - profile daily mean dollars of cash approvals |
| pddcsdec | - profile daily mean dollars of cash declines |
| pdddec | - profile daily mean dollars of declines |
| pdddec2 | - profile daily mean dollars of declines on days with auths |
| pddmrapv | - profile daily mean dollars of merchandise approvals |
| pddmrdec | - profile daily mean dollars of merchandise declines |
| pdnapv | - profile daily mean number of approvals |
| pdnau | - profile daily mean number of auths on days with auths |
| pdnau30 | - profile daily mean number of auths on all days in month |
| pdncsapv | - profile daily mean number of cash approvals |
| pdncsdec | - profile daily mean number of cash declines |
| pdndec | - profile daily mean number of declines |
| pdnmrapv | - profile daily mean number of merchandise approvals |
| pdnmrdec | - profile daily mean number of merchandise declines |
| pdnw1ap2 | - profile mean number of approvals on Sundays which had auths |
| pdnw1apv | - provilde mean number of approvals on Sundays (day 1 of week) |
| pdnw2ap2 | - profile mean number of approvals on Mondays which had auths |
| pdnw2apv | - profile mean number of approvals on Mondays (day 2 of week) |
| pdnw3ap2 | - profile mean number of approvals on Tuesdays which had auths |
| pdnw3apv | - profile mean number of approvals on Tuesdays (day 3 of week) |
| pdnw4ap2 | - profile mean number of approvals on Wednesdays which had auths |
| pdnw4apv | - profile mean number of approvals on Wednesdays (day 4 of week) |
| pdnw5ap2 | - profile mean number of approvals on Thursdays which had auths |
| pdnw5apv | - profile mean number of approvals on Thursdays (day 5 of week) |
| pdnw6ap2 | - profile mean number of approvals on Fridays which had auths |
| pdnw6apv | - profile mean number of approvals on Fridays (day 6 of week) |
| pdnw7ap2 | - profile mean number of approvals on Saturdays which had auths |
| pdnw7apv | - profile mean number of approvals on Saturdays (day 7 of week) |
| pdnwdaft | - profile daily mean number of weekday afternoon approvals |
| pdnwdapv | - profile daily mean number of weekday approvals |
| pdnwdeve | - profile daily mean number of weekday evening approvals |
| pdnwdmor | - profile daily mean number of weekday morning approvals |
| pdnwdnit | - profile daily mean number of weekday night approvals |

pdnweaft - profile daily mean number of weekend afternoon approvals

pdnweapv - profile daily mean number of weekend approvals

pdnweeve - profile daily mean number of weekend evening approvals

pdnwemor - profile daily mean number of weekend morning approvals

pdnwenit - profile daily mean number of weekend night approvals

pexpir - profile expiry date stored in profile; update if curr date>pexpir

phibal - profile highest monthly balance

phidcapv - profile highest dollar amt on a single cash approve in a month

phidcdec - profile highest dollar amt on a single cash decline in a month

phidmapv - profile highest dollar amt on a single merch approve in a month

phidmdec - profile highest dollar amt on a single merch decline in a month

phidsapv - profile highest dollar amount on a single approve in a month

phidsau - profile highest dollar amount on a single auth in a month

phidsdec - profile highest dollar amount on a single decline in a month

phidtapv - profile highest total dollar amount for an approve in a single day

phidtau - profile highest total dollar amount for any auth in a single day

phidtdec - profile highest total dollar amount for a decline in a single day

phinapv - profile highest number of approves in a single day

phinau - profile highest number of auths in a single day

phindec - profile highest number of declines in a single day

pm1avbal - profile average bal. during 1st 10 days of mo.

pm1nauths - profile number of auths in the 1st 10 days of mo.

pm2avbal - profile average bal. during 2nd 10 days of mo.

pm2nauths - profile number of auths in the 2nd 10 days of mo.

pm3avbal - profile average bal. during remaining days

pm3nauths - profile number of auths in the last part of the month.

pmovewt - profile uses last zip to determine recent residence move; pmovewt=2 for a move within the previous calendar month; pmovew

pnaudy - profile number of days with auths

pnauw1 - profile number of Sundays in month with any auths

pnauw2 - profile number of Mondays in month with any auths

pnauw3 - profile number of Tuesdays in month with any auths

pnauw4 - profile number of Wednesdays in month with any auths

pnauw5 - profile numberof Thursdays in month with any auths

pnauw6 - profile number of Fridays in month with any auths

pnauw7 - profile number of Saturdays in month with any auths

pnauwd - profile number of weekday days in month with any auths

| | |
|---|---|
| pnauwe | - profile number of weekend days in month with any auths |
| pncsaudy | - profile number of days in month with cash auths |
| pnmraudy | - profile number of days in month with merchant auths |
| pnweekday | - profile number of weekday days in the month |
| pnweekend | - profile number of weekend days in the month |
| pratdcau | - profile ratio of declines to auths |
| profage | - profile number of months this account has had a profile (up to 6 mo.) |
| psdaudy | - profile standard dev. of # days between transactions in a month |
| psddau | - profile standard dev. of $ per auth in a month |
| ptdapv | - profile total dollars of approvals in a month |
| ptdau | - profile total dollars of auths in a month |
| ptdaudy | - profile total dollars of auths in a day |
| ptdcsapv | - profile total dollars of cash advance approvals in a month |
| ptdcsdec | - profile total dollars of cash advance declines in a month |
| ptddec | - profile total dollars of declines in a month |
| ptdmrapv | - profile total dollars of merchandise approvals in a month |
| ptdmrdec | - profile total dollars of merchandise declines in a month |
| ptdsfa01 | - profile total dollars of transactions in SIC factor group 01 |
| ptdsfa02 | - profile total dollars of transactons in SIC factor group 02 |
| ptdsfa03 | - profile total dollars of transactions in SIC factor group 03 |
| ptdsfa04 | - profile total dollars of transactions in SIC factor group 04 |
| ptdsfa05 | - profile total dollars of transactions in SIC factor group 05 |
| ptdsfa06 | - profile total dollars of transactions in SIC factor group 06 |
| ptdsfa07 | - profile total dollars of transactions in SIC factor group 07 |
| ptdsfa08 | - profile total dollars of transactions in SIC factor group 08 |
| ptdsfa09 | - profile total dollars of transactions in SIC factor group 09 |
| ptdsfa10 | - profile total dollars of transactions in SIC factor group 10 |
| ptdsfa11 | - profile total dolalrs of transactions in SIC factor group 11 |
| ptdsra01 | - profile total dollars of transactions in SIC fraud rate group 01 |
| ptdsra02 | - profile total dollars of transactions in SIC fraud rate group 02 |
| ptdsra03 | - profile total dollars of transactions in SIC fraud rate group 03 |
| ptdsra04 | - profile total dollars of transactions in SIC fraud rate group 04 |
| ptdsra05 | - profile total dollars of transactions in SIC fraud rate group 05 |
| ptdsra06 | - profile total dollars of transactions in SIC fraud rate group 06 |
| ptdsra07 | - profile total dollars of transactions in SIC fraud rate group 07 |
| ptdsva01 | - profile total dollars in SIC VISA group 01 |
| ptdsva02 | - profile total dollars in SIC VISA group 02 |

| | |
|---|---|
| ptdsva03 | - profile total dollars in SIC VISA group 03 |
| ptdsva04 | - profile total dollars in SIC VISA group 04 |
| ptdsva05 | - profile total dollars in SIC VISA group 05 |
| ptdsva06 | - profile total dollars in SIC VISA group 06 |
| ptdsva07 | - profile total dollars in SIC VISA group 07 |
| ptdsva08 | - profile total dollars in SIC VISA group 08 |
| ptdsva09 | - profile total dollars in SIC VISA group 09 |
| ptdsva10 | - profile total dollars in SIC VISA group 10 |
| ptdsva11 | - profile total dollars in SIC VISA group 11 |
| ptnapv | - profile total number of approvals in a month |
| ptnapvdy | - profile total number of approves a day |
| ptnau | - profile total number of auths in a month |
| ptnau10d | - profile number of auths in month<=$10 |
| ptnaudy | - profile total number of auths in a day |
| ptncsapv | - profile total number of cash advance approvals in a month |
| ptncsdec | - profile total number of cash advance declines in a month |
| ptndec | - profile total number of declines in a month |
| ptndecdy | - profile total number of declines in a day |
| ptnmrapv | - profile total number of merchandise approvals in a month |
| ptnmrdec | - profile total number of merchandize declines in a month |
| ptnsfa01 | - profile total number of transactions in SIC factor group 01 |
| ptnsfa01 | - profile total number of transactions in SIC factor group 02 |
| ptnsfa03 | - profile total number of transactions in SIC factor group 03 |
| ptnsfa04 | - profile total number of transactions in SIC factor group 04 |
| ptnsfa05 | - profile total number of transactions in SIC factor group 05 |
| ptnsfa06 | - profile total number of transactions in SIC factor group 06 |
| ptnsfa07 | - profile total number of transactions in SIC factor group 07 |
| ptnsfa08 | - profile total number of transactions in SIC factor group 08 |
| ptnsfa09 | - profile total number of transactions in SIC factor group 09 |
| ptnsfa10 | - profile total number of transactions in SIC factor group 10 |
| ptnsfa11 | - profile total number of transactions in SIC factor group 11 |
| ptnsra01 | - profile total number of transactions in SIC fraud rate group 01 |
| ptnsra02 | - profile total number of transactions in SIC fraud rate group 02 |
| ptnsra03 | - profile total number of transactions in SIC fraud rate group 03 |
| ptnsra04 | - profile total number of transactions in SIC fraud rate group 04 |
| ptnsra05 | - profile total number of transactions in SIC fraud rate group 05 |
| ptnsra06 | - profile total number of transactions in SIC fraud rate group 06 |

| | |
|---|---|
| ptnsra07 | - profile total number of transactions in SIC fraud rate group 07 |
| ptnsva01 | - profile total number in SIC VISA group 01 |
| ptnsva02 | - profile total number in SIC VISA group 02 |
| ptnsva03 | - profile total number in SIC VISA group 03 |
| ptnsva04 | - profile total number in SIC VISA group 04 |
| ptnsva05 | - profile total number in SIC VISA group 05 |
| ptnsva06 | - profile total number in SIC VISA group 06 |
| ptnsva07 | - profile total number in SIC VISA group 07 |
| ptnsva08 | - profile total number in SIC VISA group 08 |
| ptnsva09 | - profile total number in SIC VISA group 09 |
| ptnsva10 | - profile total number in SIC VISA group 10 |
| ptnsva 11 | - profile total number in SIC VISA group 11 |
| ptnw1apv | - profile total number of approvals on Sundays (day 1 of week) |
| ptnw2apv | - profile total number of approvals on Mondays (day 2 of week) |
| ptnw3apv | - profile total number of approvals on Tuesdays (day 3 of week) |
| ptnw4apv | - profile total number of approvals on Wednesdays (day 4 of week) |
| ptnw5apv | - profile total number of approvals on Thursdays (day 5 of week) |
| ptnw6apv | - profile total number of approvals on Fridays (day 6 of week) |
| ptnw7apv | - profile total number of approvals on Saturdays (day 7 of week) |
| ptnwdaft | - profile total number of weekday afternoon approvals in a month |
| ptnwdapv | - profile total number of weekday approvals in a month |
| ptnwdeve | - profile total number of weekday evening approvals in a month |
| ptnwdmor | - profile total number of weekday morning approvals in a month |
| ptnwdnit | - profile total number of weekday night approvals in a month |
| ptnweaft | - profile total number of weekend afternoon approvals in a month |
| ptnweapv | - profile total number of weekend approvals in a month |
| ptnweeve | - profile total number of weekend evening approvals in a month |
| ptnwemor | - profile total number of weekend morning approvals in a month |
| ptnwenit | - profile total number of weekend night approvals in a month |
| pvdaybtwn | - profile variance in number of days between trx's (min of 3 trx) |
| pvraudl | - profile variance of dollars per auth across month |

MERCHANT FRAUD VARIABLES

| | |
|---|---|
| mtotturn | Merchant Total turnover for this specific merchant |
| msicturn | Merchant Cumulative SIC code turnover |
| mctrtage | Merchant Contract age for specific merchant |

| | |
|---|---|
| maagsic | Merchant Average contract age for this SIC code |
| mavgnbtc | Merchant Average number of transactions in a batch |
| maamttrx | Merchant Average amount per transaction (average amount per authorization) |
| mvaramt | Merchant Variance of amount per transaction |
| mavgtbtc | Merchant Average time between batches |
| mavgtaut | Merchant Average time between authorizations for this merchant |
| mratks | Merchant Ratio of keyed versus swiped transactions |
| mnidclac | Merchant Number of identical customer accounts |
| mnidcham | Merchant Number of identical charge amounts |
| mtrxsrc | Merchant What is the source of transaction (ATM, merchant, etc.) |
| mtrxtrsp | Merchant How is the transaction transported to the source (terminal, non-terminal, voice authorization) |
| mfloor | Merchant Floor limit |
| mchgbks | Merchant Charge-backs received |
| mrtrvs | Merchant Retrievals received (per SIC, merchant, etc.). The issuer pays for a retrieval. |
| macqrat | Merchant Acquirer risk management rate (in Europe one merchant can have multiple acquirers, but they dont have records about how many or who.) |
| mprevrsk | Merchant Previous risk management at this merchant? Yes or No |
| mtyprsk | Merchant Type of previous risk management (counterfeit, multiple imprint, lost/stolen/not received) |
| msicrat | Merchant SIC risk management rate |
| mpctaut | Merchant Percent of transactions authorized |

Network Training: Once pre-processing is complete, the fraud-related variables are fed to the network and the network is trained. The preferred embodiment uses a modeling technique known as a "feed forward" neural network. This type of network estimates parameters which define relationships among variables using a training method. The preferred training method, well known to those skilled in the art, is called "backpropagation gradient descent optimization", although other well-known neural network training techniques may also be used.

One problem with neural networks built with conventional backpropagation methods is insufficient generalizability. Generalizability is a measure of the predictive value of a neural network. The attempt to maximize generalizability can be interpreted as choosing a network model with enough complexity so as not to underfit the data but not too much complexity so as to overfit the data. One measure of the complexity of a network is the number of hidden processing elements, so that the effort to maximize generalizability translates into a selection among models having different numbers of hidden processing elements. Unfortunately, it is often not possible to obtain all the nonlinearity required for a problem by adding hidden processing elements without introducing excess complexity. Many weights that come with the addition of each new hidden processing element may not be required or even helpful for the modeling task at hand. These excess weights tend to make the network fit the idiosyncrasies or "noise" of the data and thus fail to generalize well to new cases. This problem, known as overfitting, typically arises because of an excess of weights.

Weight decay is a method of developing a neural network that minimizes overfitting without sacrificing the predictive power of the model. This method initially provides the network with all the nonlinearity it needs by providing a large number of hidden processing elements. Subsequently, it decays all the weights to varying degrees so that only the weights that are necessary for the approximation task remain. Two central premises are employed: 1) when given two models of equivalent performance on a training data set, favor the smaller model; and 2) implement a cost function that penalizes complexity as part of the backpropagation algorithm. The network is trained by minimizing this cost function. Complexity is only justified as it expresses information contained in the data. A weight set that embodies all or almost all of the information in the data and none of the noise will maximize generalizability and performance.

The cost function is constructed by introducing a "decay term" to the usual error function used to train the network. It is designed to optimize the model so that the network captures all the important information in the training set, but does not adapt to noise or random characteristics of the training set. In view of these requirements, the cost function must take into account not only prediction error, but also the significance of model weights. A combination of these two terms yields an objective function which, when minimized, generalizes optimally. Performing a conventional gradient descent with this objective function optimizes the model.

In introducing the decay term, an assumption is made about what constitutes information. The goal is to choose a decay term that accurately hypothesizes the prior distribution of the weights. In finding a good prior distribution, one examines the likelihood that the weights will have a given distribution without knowledge of the data.

Weigend et al, "Generalization by Weight-Elimination with Application to Forecasting", in Advances in Neural Information Processing Systems 3, pp. 875-82, and incorporated herein by reference, discloses the following cost function for weight decay:

$$\frac{1}{2}\sum_{k \in D}(target_k - output_k)^2 + \lambda \sum_{i \in W} \frac{\omega_i^2/\omega_o^2}{1+\omega_i^2/\omega_o^2} \qquad \text{(Eq. 1)}$$

where:

$D$ is the data set;

$target_k$ is the target, or desired, value for element $k$ of the data set;

*$output_k$* is the network output for element $k$ of the data set;
$l$ represents the relative importance of the complexity term;
$W$ is the weight set;
$w_i$ is the value of weight $i$; and
$w_o$ is a constant that controls the shape of the curve that penalizes the weights.

The first term of the Weigend function measures the performance of the network, while the second term measures the complexity of the network in terms of its size. With this cost function, small weights decay rapidly, while large weights decay slowly or not at all.

A major failing of the Weigend cost function, and similar weight decay schemes, is that they do not accurately mimic the intended prior distribution. Finding a good prior distribution (or "prior") is a key element to developing an effective model. Most of the priors in the literature are sufficient to demonstrate the concept of weight decay but lack the strengths required to accommodate a wide range of problems. This occurs because the priors tend to decay weights evenly for a given processing element, without sufficiently distinguishing important weights (which contain more information) from unimportant weights (which contain less information). This often results either in 1) undesired decaying of important weights, which diminishes the power of the system to accommodate nonlinearity, or 2) undesired retention of excess unimportant weights, which leads to overfitting.

The present invention uses the following improved cost function, which addresses the above problems:

$$\frac{1}{2}\sum_{k\in D}(target_k - output_k)^2 + gl\sum_{i\in W}(c_l w_i^2 - \frac{1}{1+|w_i|}) \qquad \text{(Eq. 2)}$$

where $g$ represents a new term known as the interlayer gain multiplier for the decay rate, and $c_l$ is a constant. The interlayer gain multiplier takes into account the relative proximity of the weights to the input and output ends of the network. Thus, the interlayer gain multiplier allows application of the decay term with greater potency to elements that are closer to the inputs, where the majority of the weights typically reside, while avoiding excessive decay on weights corresponding to elements closer to the outputs, which are more critical, since their elimination can effectively sever large numbers of input-side weights.

By intensifying decay on input-side elements, the cost function of Equation 2 improves the ability of model development component 801 to decay individual weights while preserving processing elements containing valuable information. The result is that weak interactions are eliminated while valid interactions are retained. By retaining as many processing elements as possible, the model does not lose the power to model nonlinearities, yet the overfitting problem is reduced because unnecessary individual weights are removed.

Once the cost function has been iteratively applied to the network, weights that have been decayed to a very small number (defined as e) are removed from the network. This step, known as "thresholding the net" is performed because it is often difficult to completely decay weights to zero.

Once the network has been trained using past data, the network's model definition is stored in data files. One portion of this definition, called the "CFG" file, specifies the parameters for the network's input variables, including such information as, for example, the lengths of the variables, their types, and their ranges. Referring now to Figure 21, there is shown a portion of a typical CFG file, specifying parameters for an ACCOUNT variable 2101 (representing a customer account number) and a PAUDYMDY variable 2102 (a profile variable representing the ratio of transaction days divided by the number of days in the month).

The file formats used to store the other model definition files for the network are shown below.

A

**ASCII File Formats**

The ASCII network data files (*.cta*, *.sta*, *.lca*, *.wta*) consist of tokens (non-whitespace) separated by whitespace (space, tab, newline).
Whitespace is ignored except to separate tokens. Use of line breaks and tabs is encouraged for clarity, but otherwise irrelevant.

File format notation is as follows:

* Bracketed text denotes a token.
* Nonbracketed text denotes a literal token which must be matched exactly, including case.
* Comments on the right are <u>not</u> part of the file format; they simply provide further description of the format.
* In the comments, vertical lines denote a block which can be repeated. Nested vertical lines denote repeatable sub-blocks.

*.cta* Format

| File format | Comments |
|---|---|
| `cts` | |
| `<NetName>` | |
| `<Value>` | \| Repeated as needed |

`cts` and `<NetName>` must appear first. `<NetName>` is the standard abbreviation, lowercase (e.g., `mbpn`). The `<Value>`s are the network constants values, in the order defined within the constants structure. If a constants value is an array or structured type, each element or field must be a separate token, appearing in the proper order.

| Example | Comments |
|---|---|
| `cts` | |
| `mbpn` | |
| `2` | InputSize |
| `1` | OutputSize |
| `1` | cHidSlabs |
| `2` | HiddenSize[0] |
| `0` | HiddenSize[1] |
| `0` | HiddenSize[2] |
| `3` | RandomSeed |
| `1.0` | InitWeightMax |
| `0` | WtsUpdateFlag |
| `0` | ConnectInputs |
| `0` | FnClass |
| `1.0` | Parm1 |
| `1.0` | Parm2 |
| `-1.0` | Parm3 |
| `0.0` | Parm4 |
| `0.0` | Parm5 |
| `1` | cEntTbl |
| `0.0` | xLow |
| `0.1` | xHigh |
| `0.2` | HiddenAlpha[0] |
| `0.0` | HiddenAlpha[1] |

| Example | Comments |
|---|---|
| 0.0 | HiddenAlpha[2] |
| 0.1 | OutputAlpha |
| 0.9 | HiddenBeta[0] |
| 0.0 | HiddenBeta[1] |
| 0.0 | HiddenBeta[2] |
| 0.9 | OutputBeta |
| 0.0 | Tolerance |
| 0 | WtsUpdateFlag |
| 0 | BatchSize |
| 0 | LinearOutput |
| 0 | ActTblFlag |
| 1 | StatsFlag |
| 1 | LearnFlag |

In this example, HiddenSize, HiddenAlpha, and HiddenBeta are all arrays, so each element (0, 1, 2) has a separate token, in the order they appear in the type.

*.sta* Format

| File format | Comments |
|---|---|
| sts | |
| <NetName> | |
| <cSlab> | |
| <nSlab> | \| Repeated cSlab times |
| <cPe> | \| |
| <state> | \| \| Repeated cPe times |

sts and <NetName> must appear first. <NetName> is the standard abbreviation, lowercase. <cSlab> is a count of the slabs which have states stored in the file. The remainder of the file consists of cSlab blocks, each describing the states of one slab. The order of the slab blocks in the file is not important. <nSlab> is the slab number, as defined in the *xxx.h* file. cPe is the number of states for the slab. <state> is the value of a single state. If the state type is an array or structured type, each element or field must be a separate token, appearing in the proper order. There should be cPe <state> values in the slab block.

| <u>Example</u> | <u>Comments</u> |
|---|---|
| sts | |
| mbpn | |
| 6 | cSlab |
| 0 | nSlab -- SlabInMbpn |
| 2 | cPeIn |
| 0.0 | StsIn[0] |
| 0.0 | StsIn[1] |
| 1 | nSlab -- SlabTrnMbpn |
| 1 | cPeTrn |
| 0.0 | StsTrn[0] |
| 2 | nSlab -- SlabHid0Mbpn |
| 2 | cPeHid0 |
| 0.0 | StsHid0[0] |
| 0.0 | StsHid0[1] |
| 5 | nSlab -- SlabOutMbpn |
| 1 | cPeOut |
| 0.0 | StsOut[0] |
| 6 | nSlab -- SlabBiasMbpn |
| 1 | cPeBias |
| 1.0 | StsBias[0] |
| 7 | nSlab -- SlabStatMbpn |
| 3 | cPeStat |
| 0.0 | StsStat[0] |
| 0.0 | StsStat[1] |
| 0.0 | StsStat[2] |

*.lca* Format

```
File format              Comments
    lcl
    <NetName>
    <cSlab>
    <nSlab>              | Repeated cSlab times
    <cPe>                |
    <local>              | | Repeated cPe times
```

The *.lca* format is just like the *.sta* format except that sts is replaced by `lcl`. `lcl` and `<NetName>` must appear first. `<NetName>` is the standard abbreviation, lowercase. `<cSlab>` is a count of the slabs which have local data stored in the file. The remainder of the file consists of `cSlab` blocks, each describing the local data values of one slab. `<nSlab>` is the slab number, as defined in the *xxx.h* file. The order of the slab blocks in the file is not important. `cPe` is the number of local data values for the slab. `<local>` is the value of a single local data element. If the local data type is an array or structured type, each element or field must be a separate token, appearing in the proper order. There should be `cPe <local>` values in the slab block.

| Example | Comments |
|---|---|
| lcl | |
| mbpn | |
| 3 | cSlab |
| 2 | nSlab -- SlabHid0Mbpn |
| 2 | cPe |
| 0.0 | LclHid0[0].Error |
| 0.0 | LclHid0[0].NetInp |
| 0.0 | LclHid0[1].Error |
| 0.0 | LclHid0[1].NetInp |
| 5 | nSlab -- SlabOutMbpn |
| 1 | cPe |
| 0.0 | LclOut[0].Error |
| 0.0 | LclOut[0].NetInp |
| 7 | nSlab -- SlabStatMbpn |
| 3 | cPe |
| 0 | LclStat[0].cIter |
| 0.0 | LclStat[0].Sum |
| 0 | LclStat[1].cIter |
| 0.0 | LclStat[1].Sum |
| 0 | LclStat[2].cIter |
| 0.0 | LclStat[2].Sum |

In this example, the `<local>` values are all structured types, so each field (Error and NetInp; cIter and Sum) has a separate token, in the order they appear in the type.

.*wta* Format

```
File format          Comments
   wts
   <NetName>
   <cClass>
   <nSlab>           | Repeated cClass times
   <nClass>          |
   <cIcn>            |
   <weight>          | | Repeated cIcn times
```

wts and <NetName> must appear first. <NetName> is the standard abbreviation, lowercase. <cClass> is a count of the slab/class combinations which have weights stored in the file. The remainder of the file consists of cClass blocks, each describing the weights of one slab. The order of the class blocks in the file is not important. <nSlab> is the slab number, as defined in the *xxx.h* file. <nClass> is the class number, as defined in the *xxx.h* file. <weight> is the value of a single weight. If the weight type is an array or structured type, each element or field must be a separate token, appearing in the proper order. There should be cIcn <weight> values in the slab block.

| Example | Comments |
|---|---|
| wts | |
| mbpn | |
| 2 | cClass |
| 2 | nSlab -- SlabHid0Mbpn |
| 0 | nClass -- PeHid0MbpnFromPrev |
| 6 | cIcn |
| 0.0 | WtsHid0[PE_0][0] |
| 0.0 | WtsHid0[PE_0][1] |
| 0.0 | WtsHid0[PE_0][2] |
| 0.0 | WtsHid0[PE_1][0] |
| 0.0 | WtsHid0[PE_1][1] |
| 0.0 | WtsHid0[PE_1][2] |
| 5 | nSlab -- SlabOutMbpn |
| 0 | nClass -- PeOutMbpnFromPrev |
| 3 | cIcn |
| 0.0 | WtsOut[PE_0][0] |

| Example | Comments |
|---|---|
| 0.0 | WtsOut[PE_0][1] |
| 0.0 | WtsOut[PE_0][2] |

Weights values for a slab and class are stored as a one-dimensional array, but conceptually are indexed by two values -- PE and interconnect within PE. The values are stored in row-major order, as exemplified here.

Transaction Processing Component 802

Once the model has been created, trained, and stored, fraud detection may begin. Transaction processing component 802 of system 100 preferably runs within the context of a conventional authorization or posting system for customer transactions. Transaction processing component 802 reads current transaction data and customer data from databases 805, 806, and generates as output fraud scores representing the likelihood of fraud for each transaction. Furthermore, transaction processing component 802 can compare the likelihood of fraud with a predetermined threshold value, and flag transactions for which the threshold is exceeded.

The current transaction data from database 805 typically includes information such as: transaction dollar amount; date; time (and time zone if necessary); approve/decline code; cash/merchandise code; available credit (or balance); credit line; merchant category code; merchant ZIP code; and PIN verification (if applicable).

The customer data from database 806 typically includes information from three sources: 1) general information on the customer; 2) data on all approved or declined transactions in the previous seven days; and 3) a profile record which contains data describing the customer's transactional pattern over the last six months. The general information on the customer typically includes information such as: customer ZIP code; account open date; and expiration date. The profile record is a single record in a profile database summarizing the customer's transactional pattern in terms of moving averages. The profile record is updated periodically (usually monthly) with all of the transactions from the period for the customer, as described below.

System 100 can operate as either a batch, semi-real-time, or real-time system. The structure and processing flow of each of these variations will now be described.

Batch System: Figure 14 shows operation of a batch system. Transactions are recorded throughout the day or other convenient period 1402. At the end of the day, the system performs steps 1403 to 1409 for each transaction. It obtains data describing the current transaction 1403, as well as past transaction data, customer data, and profile data 1404. It then applies this data to the neural network 1405 and obtains a fraud score 1406. If the fraud score exceeds a threshold 1407, the account is flagged 1408. In the batch system, therefore, the transaction which yielded the high fraud score cannot itself be blocked; rather, the account is flagged 1404 at the end of the day so that no future transactions are possible. Although the batch system does not permit immediate detection of fraudulent transactions, response-time constraints may mandate use of the batch system in some implementations.

Semi-Real-Time System: The semi-real-time system operates in a similar manner to the batch system and uses the same data files, but it ensures that no more than one high-scoring transaction is authorized before flagging the account. In this system, as shown in Figure 15, fraud likelihood determination is performed (steps 1504 to 1509) immediately after the transaction is authorized 1503. Steps 1504 to 1509 correspond to steps 1403 to 1409 of the batch system illustrated in Figure 14. If the likelihood of fraud is high, the account is flagged 1509 so that no future transactions are possible. Thus, as in the batch system, the current transaction cannot be blocked; however, the semi-real-time system allows subsequent transactions to be blocked.

Real-Time System: The real-time system performs fraud likelihood determination before a transaction is authorized. Because of response-time constraints, it is preferable to minimize the number of database access calls when using the real-time system. Thus, in this embodiment, all of the customer information, including general information and past transaction data, is found in a single record of profile database 806. Profile database 806 is generated from past transaction and customer data before the transaction processing component starts operating, and is updated after each transaction, as described below. Because all needed data are located in one place, the system is able to retrieve the data more quickly than in the batch or semi-real-time schemes. In order to keep the profile database 806 current, profile records are updated, using moving averages where applicable, after each transaction.

Referring now to Figure 16, there is shown a flowchart of a real-time system using the profile database. Upon receiving a merchant's request for authorization on a transaction 1602, the system obtains data for the current transaction 1603, as well as profile data summarizing transactional patterns for the customer 1604. It then applies this data to the stored neural network model 1605. A fraud score (representing the likelihood of fraud for the transaction) is obtained 1606 and compared to a threshold value 1607. Steps 1601 through 1607 occur before a transaction is authorized, so that the fraud score can be sent to an authorization system 1608 and the transaction blocked by the authorization system if the threshold has been exceeded. If the threshold is not exceeded, the low fraud score is sent to the authorization system 1609. The system then updates customer profile database 806 with the new transaction data 1610. Thus, in this system, profile database 806 is always up to date (unlike the batch and semi-real-time systems, in which profile database 806 is updated only periodically).

Referring now to Figure 12, there is shown the method of creating a profile record. The system performs the steps of this method when there is no existing profile record for the customer. The system reads the past transaction database 1101 for the past six months and the customer database 1103 (steps 1202 and 1203 respectively). It generates a new profile record

1204 with the obtained data and saves it in the profile database 1205. If there are more accounts to be processed 1206, it repeats steps 1202 through 1205.

.. Referring now to Figure 13, there is shown the method of updating an existing profile record. The system reads the past transaction database 1101 for the past six months, customer database 1103 and profile database (steps 1302, 1303, and 1304 respectively). It combines the data into a single value for each variable in the profile database. This value is generated using one of two formulas.

For variables that represent average values over a period of time (for example, mean dollars of transactions in a month), Equation 3 is used:

$$\text{newProfData} = ((1 - a) * \text{oldProfData}) + (a * \text{currentVal})) \qquad \text{(Eq. 3)}$$

For variables that represent extreme values over a period of time (for example, highest monthly balance), Equation 4 is used:

$$\text{newProfData} = \max(\text{currentVal}, b\ * \text{oldProfData}) \qquad \text{(Eq. 4)}$$

In Equations 3 and 4:

newProfData is the new value for the profile variable;

oldProfData is the old value for the profile variable;

currentVal is the most recent value of the variable, from the past transaction database; and

a and b are decay factors which are used to give more importance to recent months and less importance to months further in the past.

The value of b is set so that older data will "decay" at an acceptable rate. A typical value for b is 0.95.

The value of a is generated as follows: For the batch and semi-real-time systems, a is set to a value such that the contribution of the value from more than six months previous is nearly zero. For profiles that have been in existence for at least six months, the value of a is 1/6. For newer profiles, the value is 1/(n+1), where n is the number of months since the profile was created. For the real-time system, profile updates do not occur at regular intervals. Therefore, a is determined using the following equation:

$$a = 1 - \exp(-t/T) \qquad \text{(Eq. 5)}$$

where:

t is the time between the current transaction and the last transaction; and

T is a time constant for the specific variable.

Furthermore, for the real-time system, currentVal represents the value of the variable estimated solely using information related to the current transaction and the time since the last transaction, without reference to any other historical information.

Once the new values for the profile variables have been generated, they are placed in an updated profile record 1305 and saved in the profile database 1306. If there are more accounts to be processed 1307, the system repeats steps 1302 through 1306.

In all of these embodiments, the current transaction data and the customer data are preferably pre-processed to derive fraud-related variables which have been empirically determined to be effective predictors of fraud. This is done using the same technique and the same fraud-related variables as described above in connection with neural network training.

Referring now to Figures 17 through 19, there are shown flowcharts illustrating the operation of the preferred embodiment of the transaction processing component. Some of the individual elements of the flowchart are indicated by designations which correspond to module names.

Referring now to Figure 17, there is shown the overall operation of transaction processing component 802. First the system runs module CINITNET 1702, which initializes network structures. Then, it runs module CSCORE 1703. Module CSCORE 1703 uses current trans-action data, data describing transactions over the past seven days, a profile record, and customer data to generate a fraud score indicating the likelihood that the current transaction is fraudulent, as well as reason codes (described below). The system then checks to see whether there are more transactions to be processed 1704, and repeats module CSCORE 1703 for any additional transactions. When there are no more to be processed, the system runs module FREENET 1705, which frees the network structures to allow them to be used for further processing.

Referring now to Figure 18, there is shown the operation of module CSCORE 1703. First, module CSCORE 1703 obtains current transaction data, data describing transactions of the past seven days, the profile record, and customer data (steps 1802 through 1805). From these data, module CSCORE 1703 generates the fraud-related variables 1806 described above. Then, it runs module DeployNet 1807, which applies the fraud-related variables to the stored neural network and provides a fraud score and reason codes. CSCORE then outputs the score and reason codes 1808.

Referring now to Figure 19, there is shown the operation of module DeployNet 1807. Module DeployNet 1807 first scales the fraud-related variables 1902 to match the scaling previously performed in model development. If the value of a variable is missing, DeployNet sets the value to equal the mean value found in the training set. Then it applies the scaled variables to the input layer of neural network 108, in step 1903. In step 1904, it processes the applied data through the network to generate the fraud score. The method of iterating the network is well known in the art.

In addition to providing fraud scores, in step 1904, module DeployNet 1807 optionally generates "reason codes". These codes indicate which inputs to the model are most important in determining the fraud score for a given transaction. Any technique that can track such reasons may be used. In the preferred embodiment, the technique set forth in co-pending U.S. application Serial No. 07/814,179, (attorney's docket number 726) for "Neural Network Having Expert System Functionality", by Curt A. Levey, filed December 30, 1991, the disclosure of which is hereby incorporated by reference, is used.

The following module descriptions summarize the functions performed by the individual modules.

**FALCON C FILES**

---

FILE NAME: CINITNET
DESCRIPTION: Contains code to allocate and initialize the network structures.

FUNCTION NAME: CINITNET()
DESCRIPTION: Allocate and initialize the network structures.

---

FILE NAME: CSCORE
DESCRIPTION: Generates fraud related variables and iterates the neural network.

FUNCTION NAME: SCORE()
DESCRIPTION: Creates fraud related variables from raw variables and makes calls to initialize the input layer and iterate the neural network.

FUNCTION NAME: setInput()
DESCRIPTION: Sets the input value for a processing element in the input layer.

FUNCTION NAME: hiReason()
DESCRIPTION: Finds the three highest reasons for the score.

---

FILE NAME: CFREENET
DESCRIPTION: Makes function calls to free the network structures.

FUNCTION NAME: CFREENET()

DESCRIPTION: Frees the network structures.

---

FILE NAME: CCREATEP

DESCRIPTION: Contains the cardholder profile creation code.

FUNCTION NAME: createpf()

DESCRIPTION: Creates a profile record for a cardholder using the previous month's authorizations and cardholder data.

---

FILE NAME: CUPDATEP

DESCRIPTION: Updates a profile of individual cardholder activity.

FUNCTION NAME: updatepf()

DESCRIPTION: Updates a profile record for a cardholder using the previous profile record values as well as the previous month's authorizations and cardholder data.

---

FILE NAME: CCOMMON

DESCRIPTION: This file contains functions which are needed by at least two of the following: createpf(), updatepf(), score().

FUNCTION NAME: accumMiscCnts()

DESCRIPTION: Increments counters of various types for each authorization found.

FUNCTION NAME: accumSicCnts()

DESCRIPTION: Increments SIC variable counters.

FUNCTION NAME: initSicCounts()

DESCRIPTION: Initializes the SIC variable counters.

FUNCTION NAME: updateSicMovAvgs()

DESCRIPTION: Updates the SIC profile variables.

FUNCTION NAME: writeMiscToProfile()

DESCRIPTION: Writes various variables to the profile record after they have been calculated.

FUNCTION NAME: hncDate()
DESCRIPTION: Converts a Julian date to a date indicating the number of days since Jan. 1, 1990.

FUNCTION NAME: missStr()
DESCRIPTION: Checks for "missing" flag (a period) in a null terminated string. String must have only blanks and a period to qualify as missing. A string with only blanks will also qualify as "missing".

Cascaded Operation

One way to improve system performance is via "cascaded" operation. In cascaded operation, more than one neural network model is used. The second neural network model is trained by model development component 801 in a similar manner to that described earlier. However, in training the second model, model development component 801 uses only those transactions that have fraud scores, as determined by prior application to the first neural network model, above a predetermined cascade threshold. Thus, the second model provides more accurate scores for high-scoring transactions. While the same fraud-related variables are available to train both models, it is often the case that different variables are found to be significant in the two models.

Referring now to Figure 20, there is shown a flowchart of the operation of the transaction processing component in a cascaded system. First, transaction processing component 802 scores each transaction using the first model 2002, as described above. Those transactions that score above the cascade threshold 2003 are applied to the second neural network model 2005. The system outputs scores and reason codes from either the first model 2004 or the second model 2006, as appropriate.

The above-described cascading technique may be extended to include three or more neural network models, each having a corresponding cascade threshold.

Performance Monitor

The system periodically monitors its performance by measuring a performance metric comprising the fraud detection rate and the false positive rate. Other factors and statistics may also be incorporated into the performance metric. When the performance metric falls below a predetermined performance level, the system may either inform the user that the fraud model needs to be redeveloped, or it may proceed with model redevelopment automatically.

From the above description, it will be apparent that the invention disclosed herein provides a novel and advantageous method of detecting fraudulent use of customer accounts and account numbers, which achieves high detection rates while keeping false positive rates relatively low. The foregoing discussion discloses and describes merely exemplary methods and embodiments of the present invention. As will be understood by those familiar with the art, the invention may be embodied in many other specific forms without departing from the spirit or essential characteristics thereof. For example, other predictive modeling techniques besides neural networks might be used. In addition, other variables might be used in both the model development and transaction processing components.

Accordingly, the disclosure of the present invention is intended to be illustrative of the preferred embodiments and is not meant to limit the scope of the invention. The scope of the invention is to be limited only by the following claims.

## CLAIMS

What is claimed is:

1. A computer-implemented process for detecting fraud for a transaction on a customer account, comprising the steps of:

developing a predictive model from past transaction data;
storing the predictive model in a medium associated with the computer;
obtaining current transaction data;
obtaining customer data;
generating a signal indicative of the likelihood of fraud responsive to application of the current transaction data and the customer data to the stored predictive model.

2. The computer-implemented process of claim 1, wherein the step of obtaining customer data comprises accessing a database containing general customer data and a database containing customer transactional pattern data.

3. The computer-implemented process of claim 1, wherein the step of obtaining customer data comprises accessing no more than one profile database record containing customer transactional pattern data.

4. The computer-implemented process of claim 3, wherein the profile database record further contains general customer data.

5. The computer-implemented process of claim 1, wherein the current transaction data and the customer data each comprise a plurality of elements, the process further comprising the steps of, for each element of the current transaction data and the customer data:

determining a relative contribution of the element to the determined likelihood of fraud;
determining from each relative contribution thus determined a reason code value; and
generating a signal indicative of the reason code value.

6. The computer-implemented process of claim 1, further comprising the steps of:

comparing the determined likelihood of fraud with a preset threshold value; and
responsive to the likelihood of fraud exceeding the preset threshold value, signaling a fraud.

7. The computer-implemented process of claim 1, further comprising the iterative steps of:

comparing the determined likelihood of fraud with a cascade threshold value; and
responsive to the likelihood of fraud exceeding the cascade threshold value, generating another signal indicative of the likelihood of fraud responsive to application of the current transaction data and the customer data to another stored predictive model.

8. The computer-implemented process of claim 1, further comprising the steps of:
monitoring a performance metric of the predictive model;
comparing the performance metric with a predetermined performance level; and
developing and storing a new predictive model from past transaction data responsive to the predetermined performance level exceeding the performance metric.

9. The computer-implemented process of claim 8, wherein the performance metric comprises:
a fraud detection rate measurement; and
a false positive rate measurement.

10. The computer-implemented process of claim 1, wherein the predictive model is a neural network.

11. The computer-implemented process of claim 1, wherein the step of developing the predictive model comprises the substeps of:
obtaining the past transaction data;
pre-processing the past transaction data to derive past fraud-related variables; and
training the predictive model with the derived past fraud-related variables.

12. The computer-implemented process of claim 11, wherein the substep of training the predictive model comprises the iterative substeps of:
applying input data to the model;
ranking output data produced thereby responsive to a measure of quality; and
adjusting operation of the model responsive to the results of the ranking step.

13. The computer-implemented process of claim 12, wherein the predictive model comprises a neural network having a plurality of interconnected processing elements, each processing element comprising:
a plurality of inputs;
a plurality of weights, each associated with a corresponding input to generate weighted inputs;
means for combining the weighted inputs; and

a transfer function for processing the combined weighted inputs to produce an output.

14. The computer-implemented process of claim 13, wherein the substep of adjusting operation of the model comprises the substeps of:

selecting a subset of the weights to be decayed; and

decaying the selected subset of weights.

15. The computer-implemented process of claim 14, wherein the substep of selecting a subset of the weights to be decayed comprises applying and minimizing a cost function including an interlayer gain multiplier which varies a decay rate responsive to the location of a weight within the network.

16. The computer-implemented process of claim 15, wherein the cost function is of the form:

$$\frac{1}{2}\sum_{k\in D}(target_k - output_k)^2 + gl\sum_{i\in W}(c_l w_i^2 - \frac{1}{1+|w_i|}) ,$$

wherein:

$D$ represents a data set;

$target_k$ represents a target value for element $k$ of the data set;

$output_k$ represents a network output for element $k$ of the data set;

$g$ represents the interlayer gain multiplier;

$l$ represents the relative importance of the complexity term;

$W$ represents a weight set;

$w_i$ represents a value of weight $i$; and

$c_l$ represents a constant.

17. A computer-implemented process for detecting fraud for a transaction on a customer account, comprising the steps of:

obtaining past transaction data;

pre-processing the past transaction data to derive past fraud-related variables;

training a predictive model with the derived past fraud-related variables;

storing the predictive model in a medium associated with the computer;

obtaining current transaction data;

pre-processing the current transaction data to derive current fraud-related variables;

obtaining customer data;

pre-processing the customer data to derive customer fraud-related variables; and

generating a signal responsive to the likelihood of fraud responsive to application of the current fraud-related variables and the customer fraud-related variables to the stored predictive model.

18. The computer-implemented process of claim 17, wherein the past fraud-related variables and the current fraud-related variables each comprise at least:

factors obtained from data referring to transaction dollar amounts related to fraud;

factors obtained from data referring to transaction dates and times related to fraud;

factors obtained from data referring to transaction approvals and declines related to fraud; and

factors obtained from data referring to risk groups related to fraud.

19. The computer-implemented process of claim 17, wherein the past fraud-related variables and the current fraud-related variables each comprise at least:

factors obtained from data referring to customers related to fraud; and

factors obtained from data referring to merchants related to fraud.

20. The computer-implemented process of claim 17, further comprising the steps of, for each of a set of the derived current fraud-related variables and the derived customer fraud-related variables:

determining a relative contribution of the variable to the determined likelihood of fraud to generate a reason code value;

determining from each relative contribution thus determined a reason code value; and

generating a signal indicative of the reason code value.

21. The computer-implemented process of claim 17, wherein the predictive model is a neural network.

22. A computer-implemented process of training a predictive model, the predictive model for predicting outcomes based on selected characteristics, the predictive model stored in a medium associated with the computer, the process comprising the iterative steps of:

applying input data to the model;

ranking output data produced thereby responsive to a measure of quality; and

adjusting operation of the model responsive to the results of the ranking step.

23. A computer-implemented process of training a neural network, the neural network for predicting outcomes based on selected characteristics, the neural network stored in a medium associated with the computer and comprising a plurality of interconnected processing elements, each processing element comprising:

a plurality of inputs;

a plurality of weights, each associated with a corresponding input to generate weighted inputs;

means for combining the weighted inputs; and

a transfer function for processing the combined weighted inputs to produce an output;

the process comprising the iterative steps of:

applying input data to the neural network;

ranking output data produced thereby responsive to a measure of quality; and

adjusting operation of the neural network responsive to the results of the ranking step.

24. The computer-implemented process of claim 23, wherein the step of adjusting operation of the neural network comprises the substeps of:

selecting a subset of the weights to be decayed; and

decaying the selected subset of weights.

25. The computer-implemented process of claim 24, wherein the substep of selecting a subset of the weights to be decayed comprises applying and minimizing a cost function including an interlayer gain multiplier which varies a decay rate responsive to the location of a weight within the network.

26. The computer-implemented process of claim 25, wherein the cost function is of the form:

$$\frac{1}{2}\sum_{k\in D}(\mathrm{target}_k - \mathrm{output}_k)^2 + g\lambda\sum_{i\in W}(c_1\omega_i{}^2 - \frac{1}{1+|\omega_i|}) \ ,$$

wherein:

*D* represents a data set;

*target*$_k$ represents a target value for element *k* of the data set;

*output*$_k$ represents a network output for element *k* of the data set;

*g* represents the interlayer gain multiplier;

*l* represents the relative importance of the complexity term;

*W* represents a weight set;

$w_i$ represents a value of weight *i;* and

$c_1$ represents a constant.

27. A system for detecting fraud for a transaction on a customer account, comprising:

a predictive model for determining a likelihood of fraud for the transaction;

past transaction data input means for obtaining past transaction data;

a model development component, coupled to the predictive model, for training the prea predictive model for determining a likelihood of fraud for the transaction;

past transaction data input means for obtaining past transaction data;

a past transaction data pre-processor for deriving past fraud-related variables from the past transaction data;

a model development component, coupled to the predictive model, for training the predictive model from the past fraud-related variables;

a storage device for storing the predictive model;

current transaction data input means for obtaining current transaction data for the transaction;

a current transaction data pre-processor for deriving current fraud-related variables from the current transaction data and sending the current fraud-related variables to the predictive model;

customer data input means for obtaining customer data;

a customer data pre-processor for deriving customer fraud-related variables from the customer data and sending the customer fraud-related variables to the predictive dictive model from the past transaction data;

a storage device for storing the trained predictive model;

current transaction data input means for obtaining current transaction data and sending the current transaction data to the predictive model;

customer data input means for obtaining and sending to the predictive model, customer data; and

an output device, coupled to the predictive model, for generating a signal responsive to the likelihood of fraud.

28. The system of claim 27, wherein the model development component comprises a past transaction data pre-processor for deriving past fraud-related variables from the past 'transaction data.

29. The system of claim 27, wherein the predictive model comprises a neural network.

30. A system for detecting fraud for a transaction on an account belonging to a customer, comprising:

model; and

an output device, coupled to the predictive model, for generating a signal responsive to the likelihood of fraud.

31. The system of claim 29, wherein the predictive model comprises a neural network.

# INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 93/08400

| Patent document cited in search report | Publication date | Patent family member(s) | | Publication date |
|---|---|---|---|---|
| EP-A-0418144 | 20-03-91 | FR-A- | 2651903 | 15-03-91 |
| EP-A-0421808 | 10-04-91 | US-A- | 5175416 | 29-12-92 |

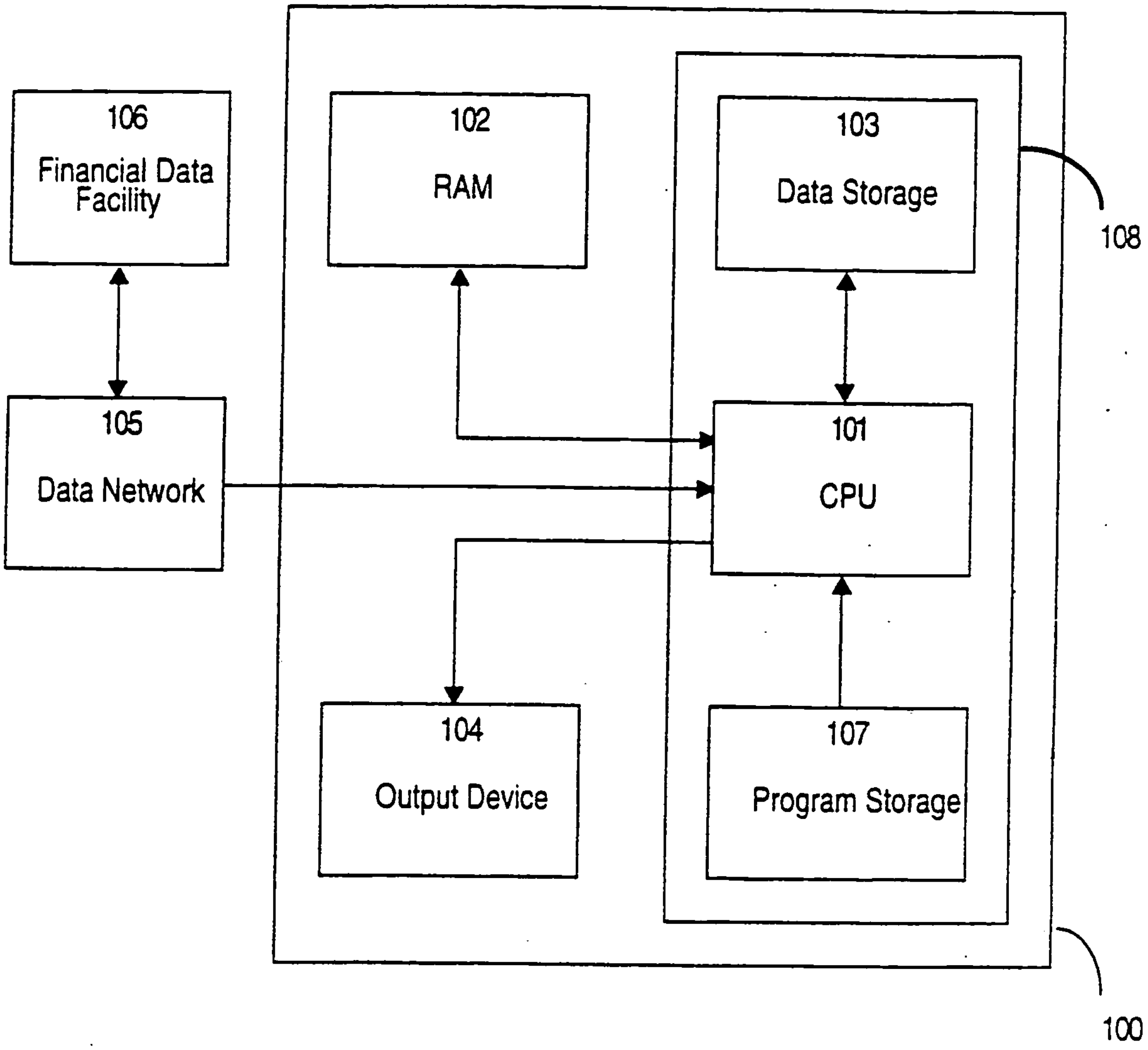
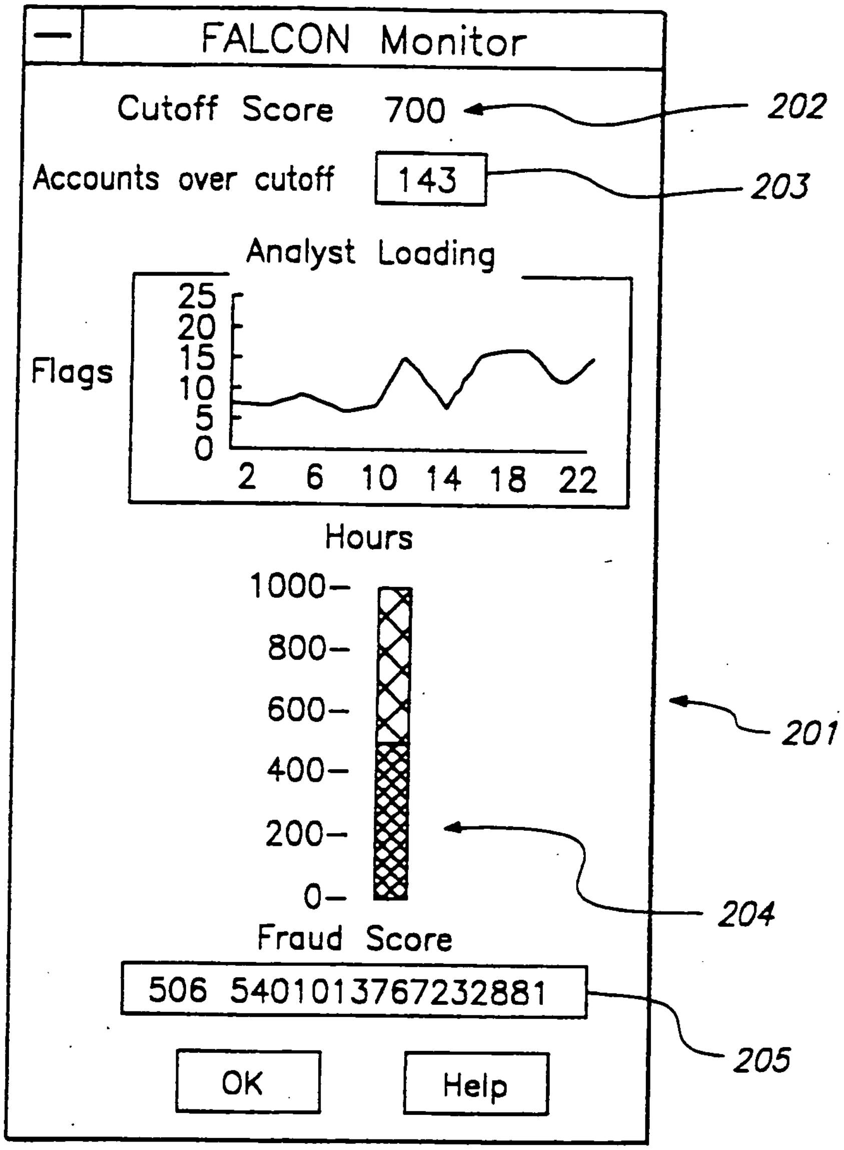

106
Financial Data Facility
105
Data Network
102
RAM
103
Data Storage
101
CPU
104
Output Device
107
Program Storage
108
100

FIGURE 1



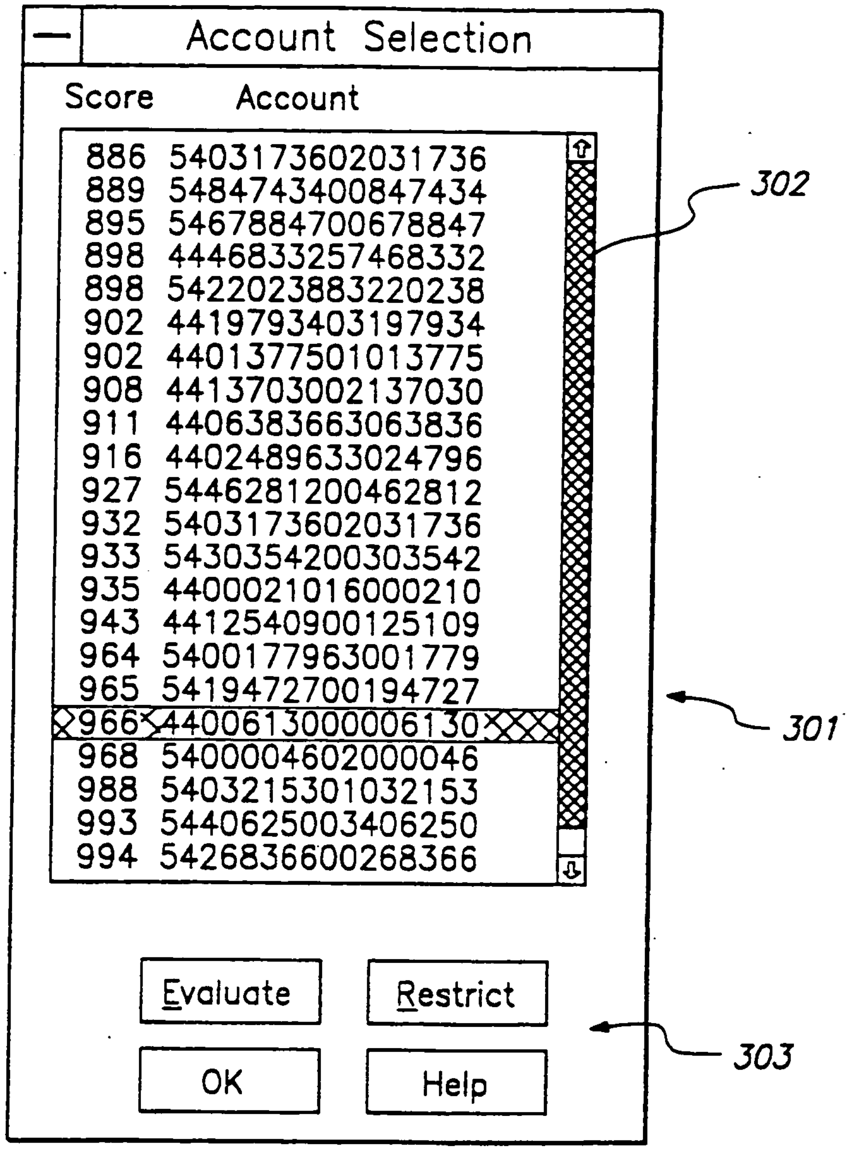
Cutoff Score 700

Accounts over cutoff 143

Fraud Score

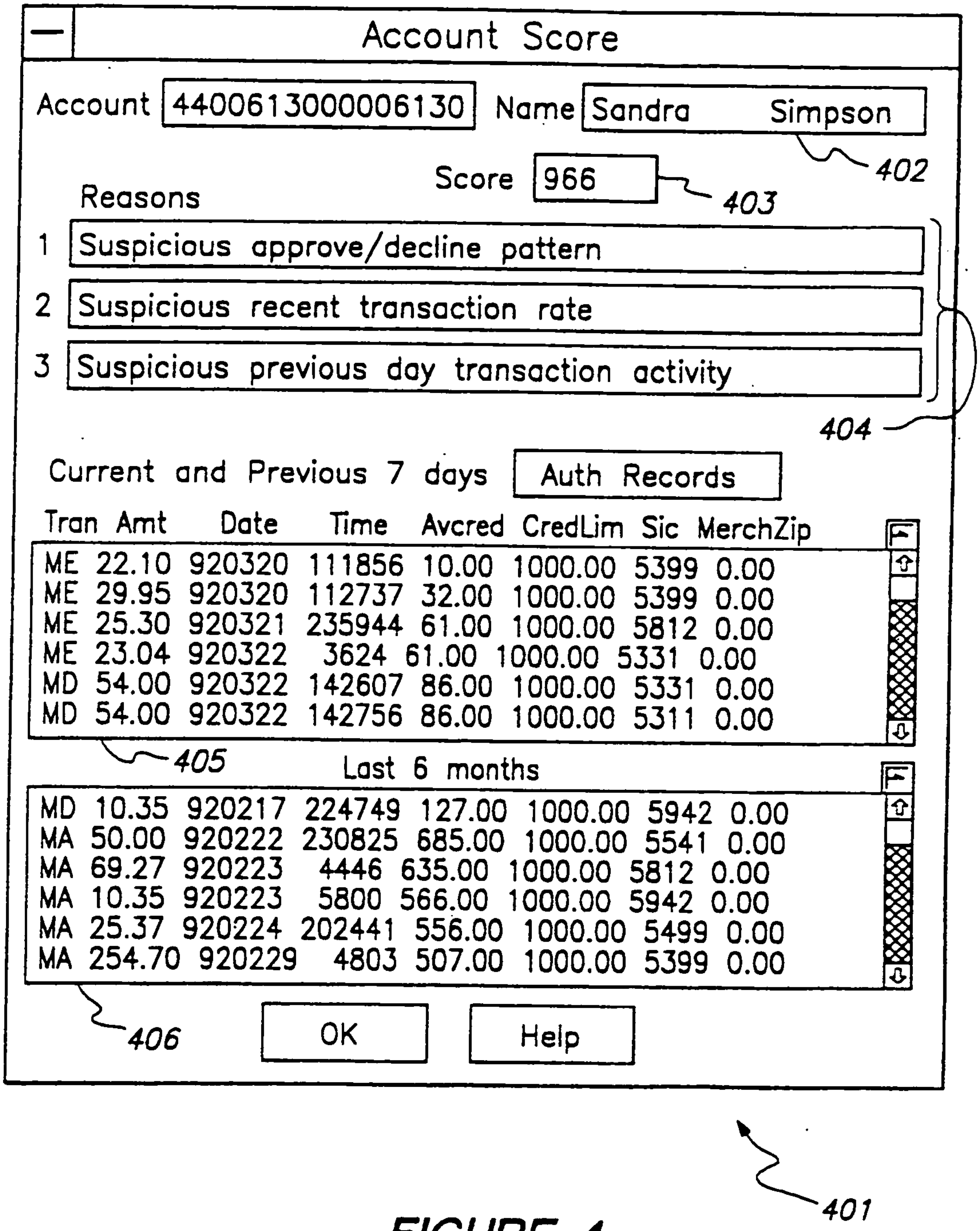
506 5401013767232881

FIGURE 2



| Score | Account |
|---|---|
| 886 | 5403173602031736 |
| 889 | 5484743400847434 |
| 895 | 5467884700678847 |
| 898 | 4446833257468332 |
| 898 | 5422023883220238 |
| 902 | 4419793403197934 |
| 902 | 4401377501013775 |
| 908 | 4413703002137030 |
| 911 | 4406383663063836 |
| 916 | 4402489633024796 |
| 927 | 5446281200462812 |
| 932 | 5403173602031736 |
| 933 | 5430354200303542 |
| 935 | 4400021016000210 |
| 943 | 4412540900125109 |
| 964 | 5400177963001779 |
| 965 | 5419472700194727 |
| 966 | 4400613000006130 |
| 968 | 5400004602000046 |
| 988 | 5403215301032153 |
| 993 | 5440625003406250 |
| 994 | 5426836600268366 |

FIGURE 3



Account Score

| | | | |
|---|---|---|---|
| Account | 4400613000006130 | Name | Sandra Simpson |
| | | Score | 966 |

Reasons

1 Suspicious approve/decline pattern

2 Suspicious recent transaction rate

3 Suspicious previous day transaction activity

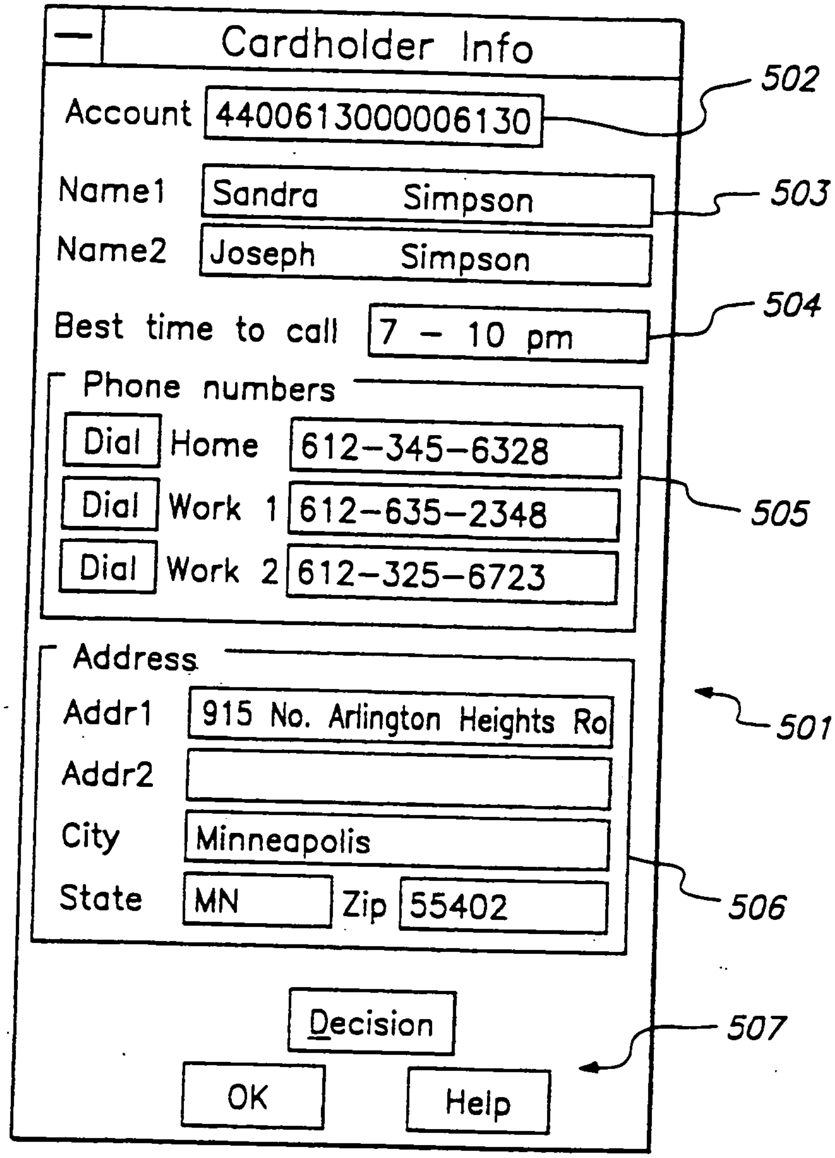
Current and Previous 7 days Auth Records

| Tran | Amt | Date | Time | Avcred | CredLim | Sic | MerchZip |
|---|---|---|---|---|---|---|---|
| ME | 22.10 | 920320 | 111856 | 10.00 | 1000.00 | 5399 | 0.00 |
| ME | 29.95 | 920320 | 112737 | 32.00 | 1000.00 | 5399 | 0.00 |
| ME | 25.30 | 920321 | 235944 | 61.00 | 1000.00 | 5812 | 0.00 |
| ME | 23.04 | 920322 | 3624 | 61.00 | 1000.00 | 5331 | 0.00 |
| MD | 54.00 | 920322 | 142607 | 86.00 | 1000.00 | 5331 | 0.00 |
| MD | 54.00 | 920322 | 142756 | 86.00 | 1000.00 | 5311 | 0.00 |

Last 6 months

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| MD | 10.35 | 920217 | 224749 | 127.00 | 1000.00 | 5942 | 0.00 |
| MA | 50.00 | 920222 | 230825 | 685.00 | 1000.00 | 5541 | 0.00 |
| MA | 69.27 | 920223 | 4446 | 635.00 | 1000.00 | 5812 | 0.00 |
| MA | 10.35 | 920223 | 5800 | 566.00 | 1000.00 | 5942 | 0.00 |
| MA | 25.37 | 920224 | 202441 | 556.00 | 1000.00 | 5499 | 0.00 |
| MA | 254.70 | 920229 | 4803 | 507.00 | 1000.00 | 5399 | 0.00 |

FIGURE 4



Cardholder Info

Account 440061300006130

Name1 Sandra Simpson

Name2 Joseph Simpson

Best time to call 7 - 10 pm

Phone numbers

Home 612-345-6328

Work 1 612-635-2348

Work 2 612-325-6723

Address

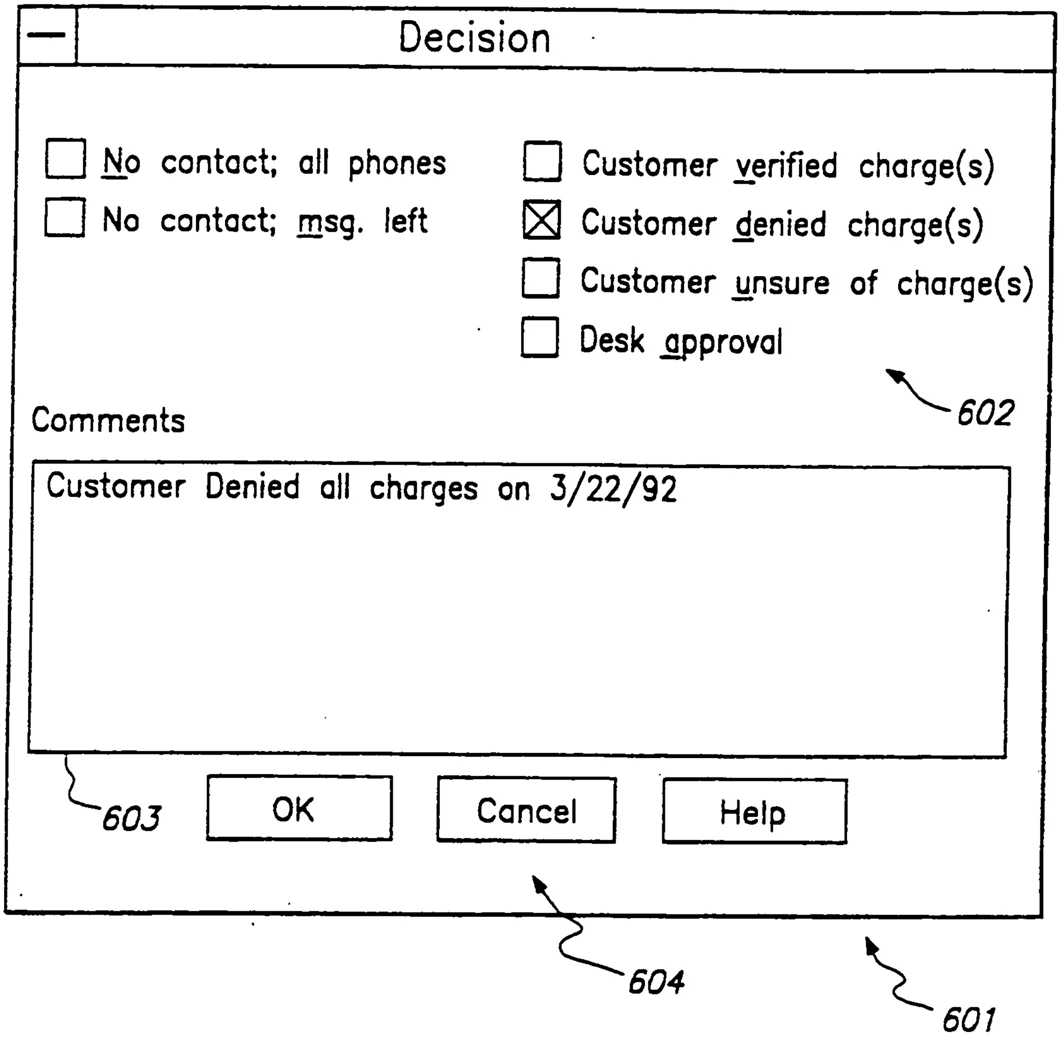
Addr1 915 No. Arlington Heights Ro

Addr2

City Minneapolis

State MN Zip 55402

FIGURE 5



Decision

☐ No contact; all phones
☐ No contact; msg. left
☐ Customer verified charge(s)
☒ Customer denied charge(s)
☐ Customer unsure of charge(s)
☐ Desk approval

Comments

Customer Denied all charges on 3/22/92

FIGURE 6

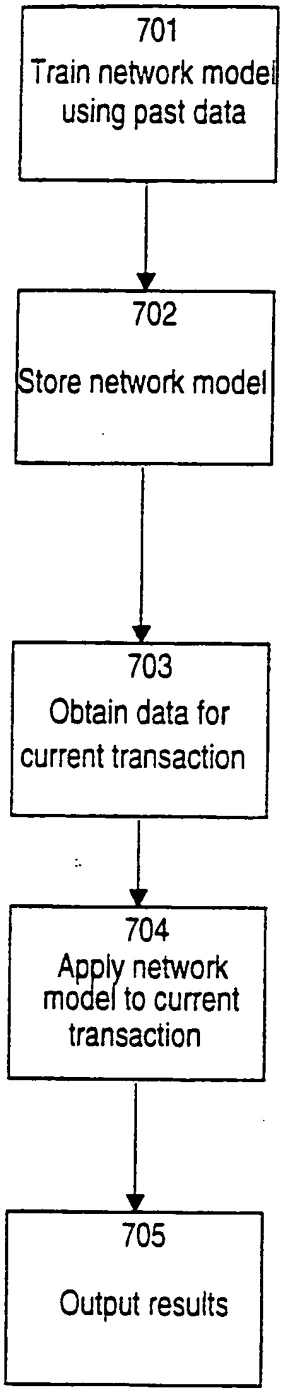

701
Train network model
using past data
702
Store network model
703
Obtain data for
current transaction
704
Apply network
model to current
transaction
705
Output results

FIGURE 7

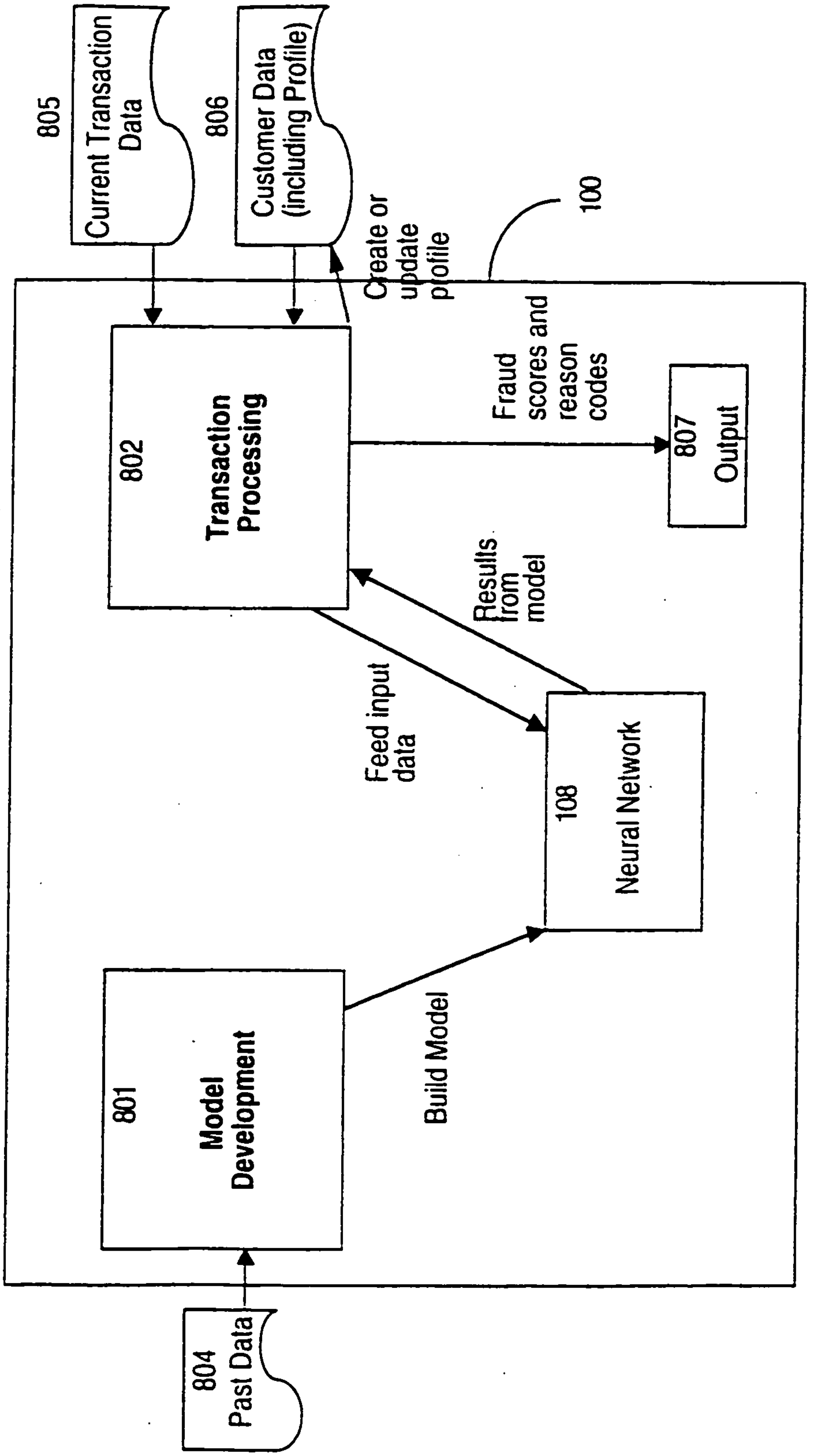

805
Current Transaction Data
806
Customer Data (including Profile)
Create or update profile
100
802
Transaction Processing
Fraud scores and reason codes
807
Output
Results from model
Feed input data
108
Neural Network
801
Model Development
Build Model
804
Past Data

FIGURE 8

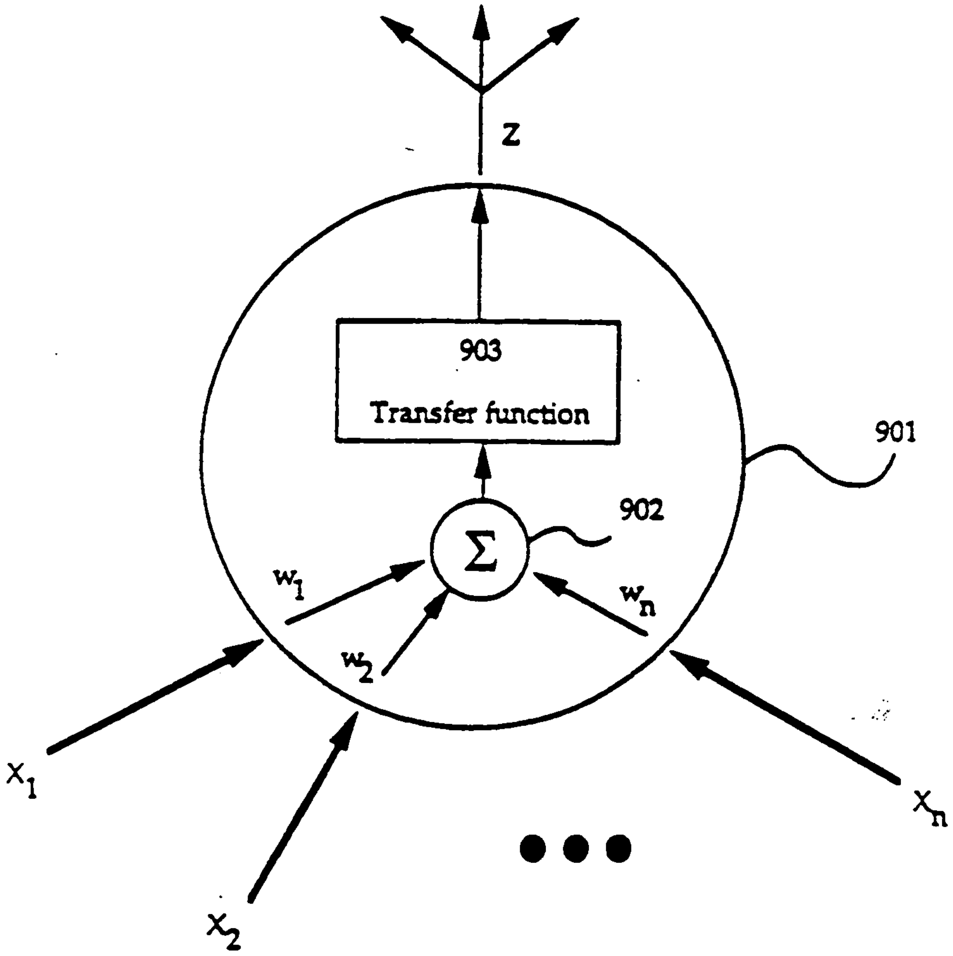


FIGURE 9

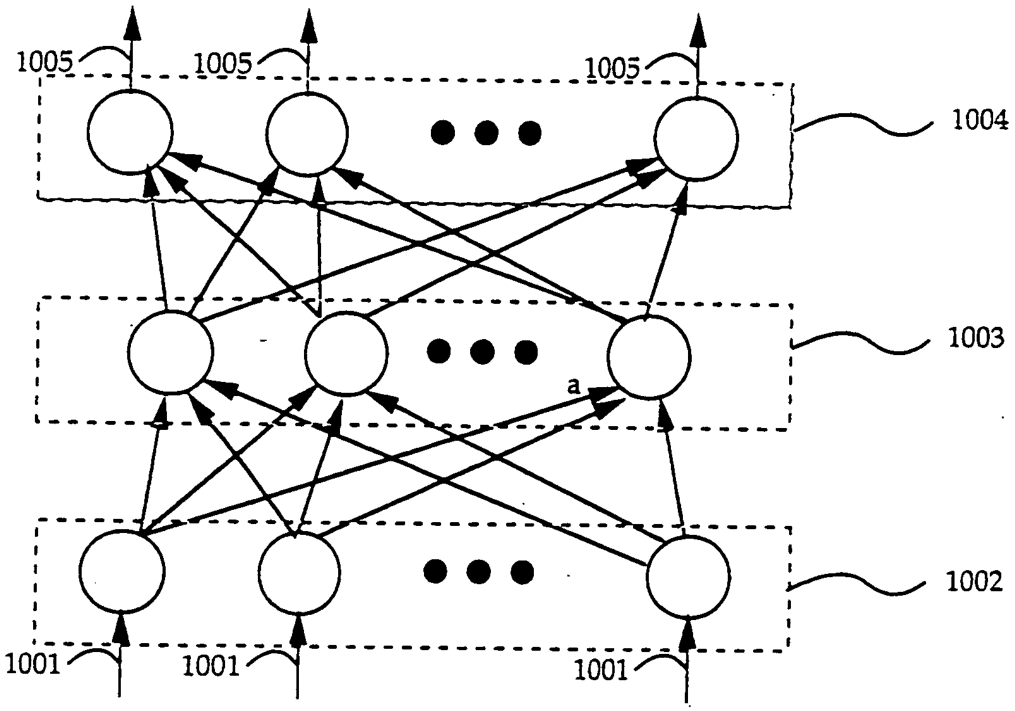


FIGURE 10

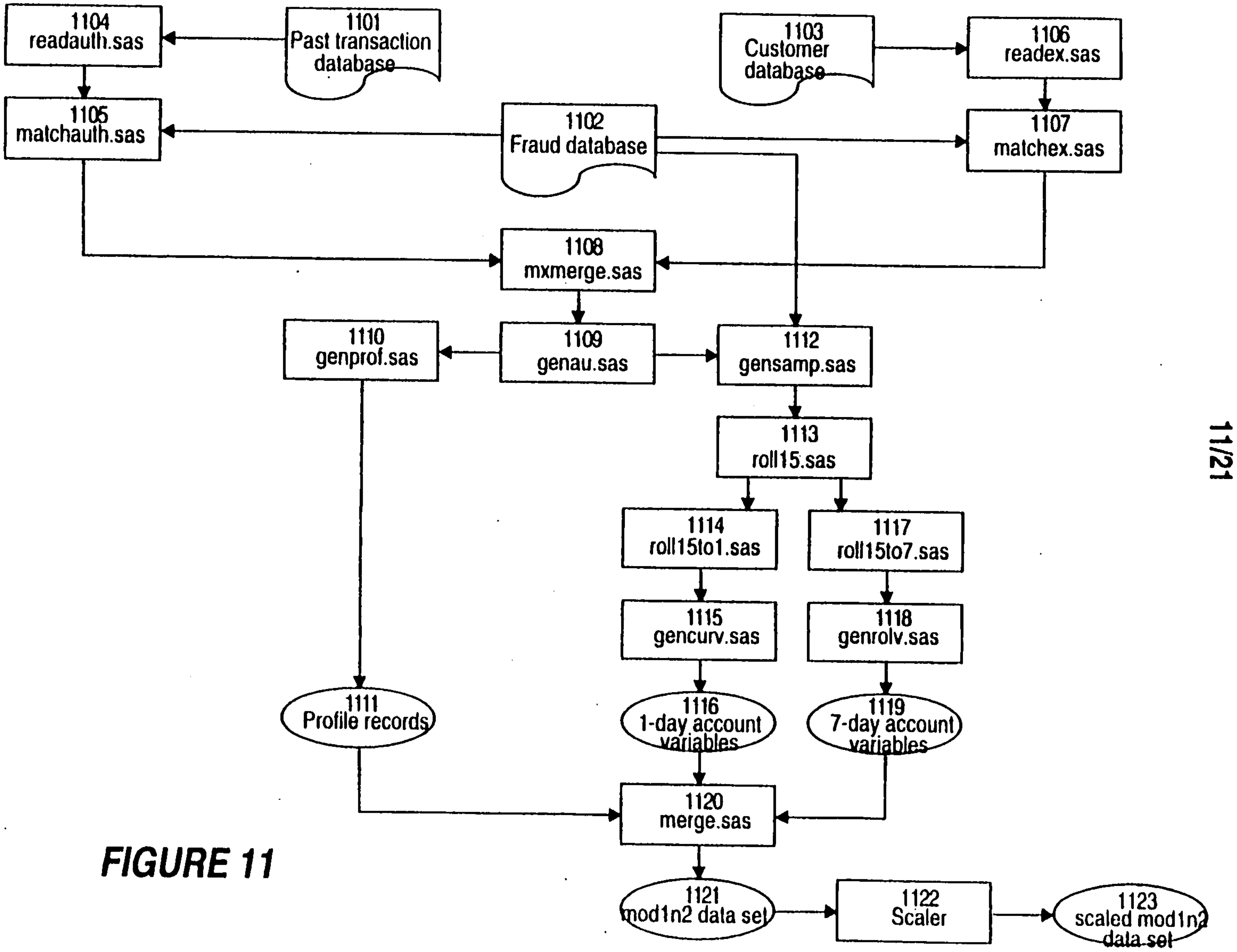

1104
readauth.sas
1101
Past transaction
database
1103
Customer
database
1106
readex.sas
1105
matchauth.sas
1102
Fraud database
1107
matchex.sas
1108
mxmerge.sas
1110
genprof.sas
1109
genau.sas
1112
gensamp.sas
1113
roll15.sas
1114
roll15to1.sas
1117
roll15to7.sas
1115
gencurv.sas
1118
genrolv.sas
1111
Profile records
1116
1-day account
variables
1119
7-day account
variables
1120
merge.sas
1121
mod1n2 data set
1122
Scaler
1123
scaled mod1n2
data set

FIGURE 11

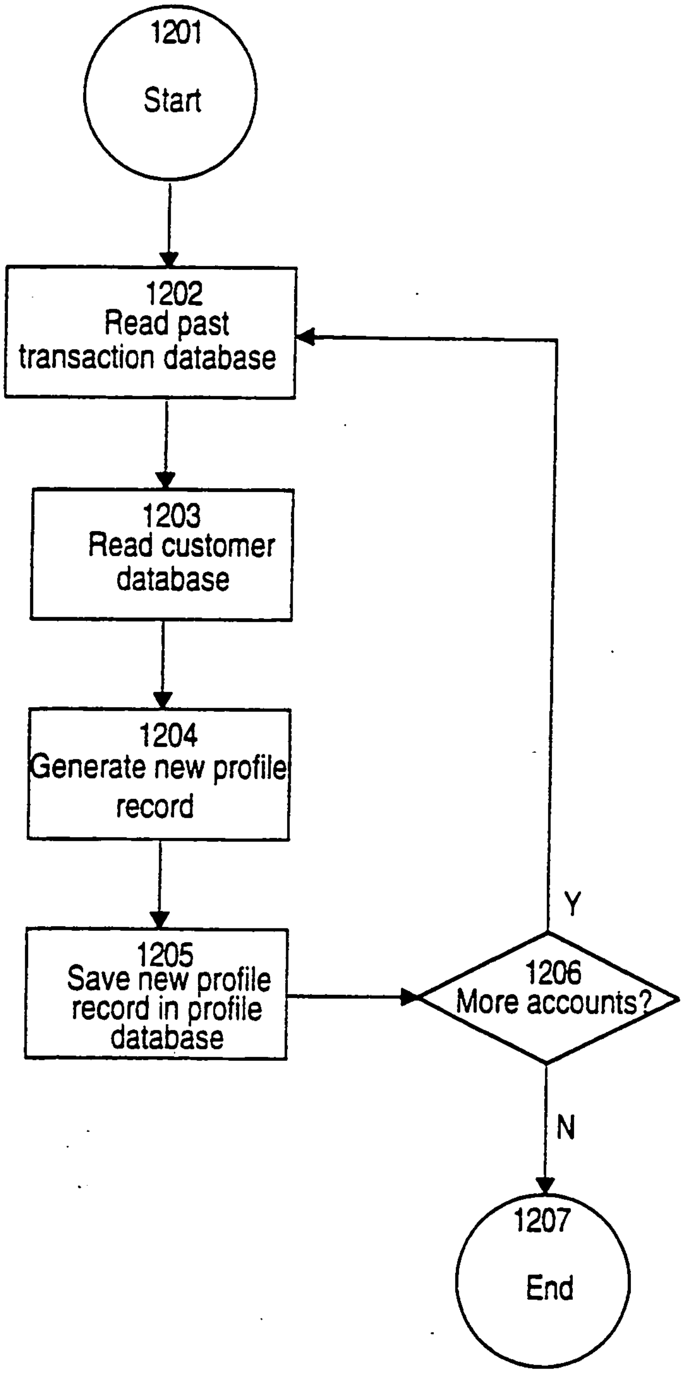

1201
Start
1202
Read past
transaction database
1203
Read customer
database
1204
Generate new profile
record
1205
Save new profile
record in profile
database
1206
More accounts?
Y
N
1207
End

FIGURE 12

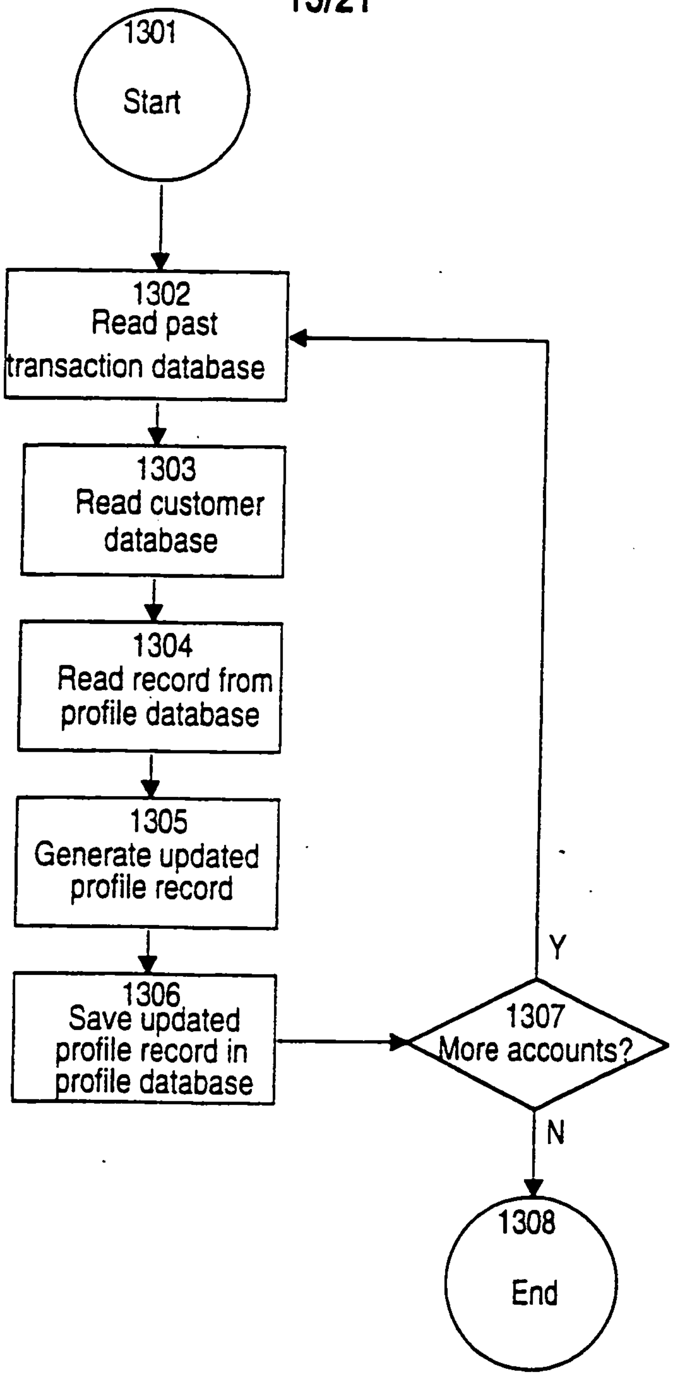

1301
Start
1302
Read past
transaction database
1303
Read customer
database
1304
Read record from
profile database
1305
Generate updated
profile record
1306
Save updated
profile record in
profile database
1307
More accounts?
Y
N
1308
End

FIGURE 13

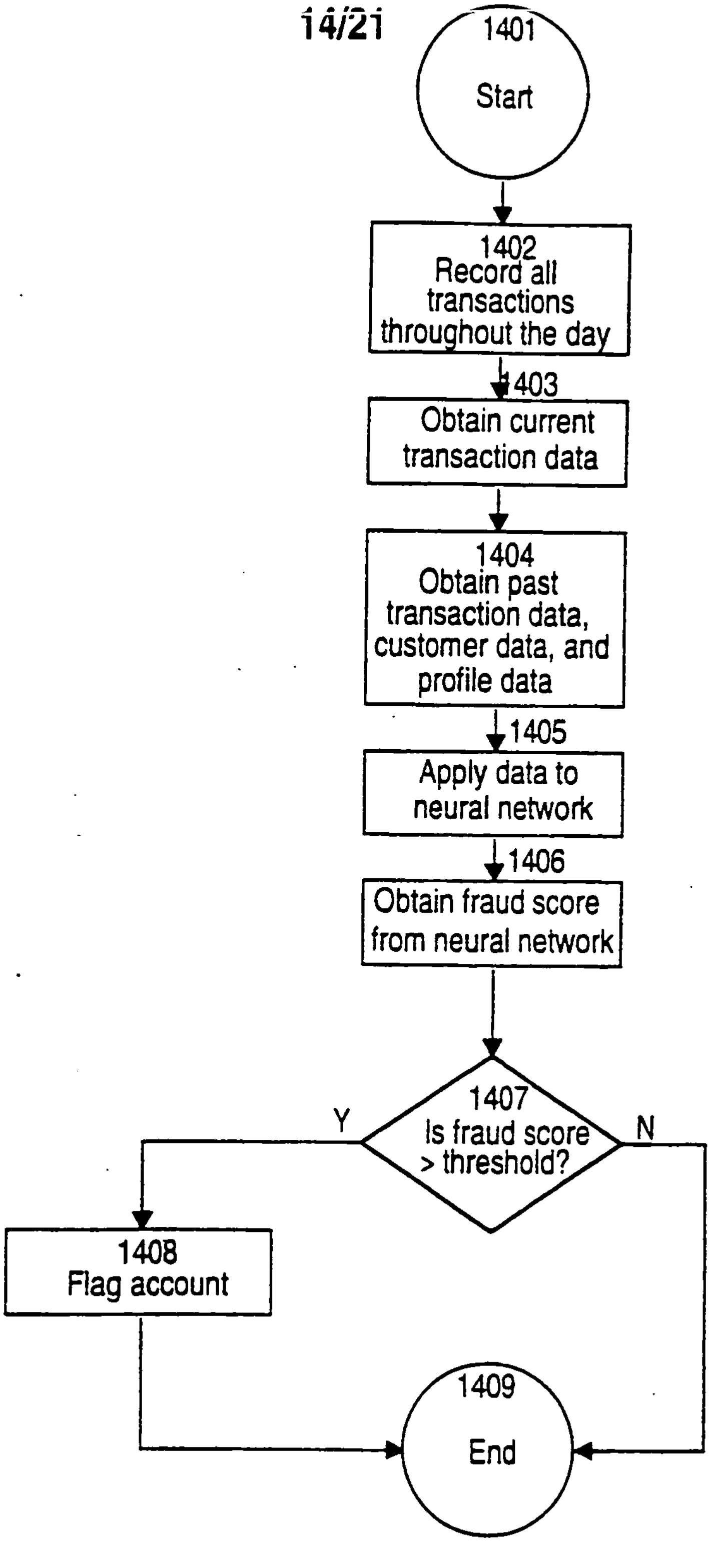

1401
Start
1402
Record all transactions throughout the day
1403
Obtain current transaction data
1404
Obtain past transaction data, customer data, and profile data
1405
Apply data to neural network
1406
Obtain fraud score from neural network
1407
Is fraud score > threshold?
Y
N
1408
Flag account
1409
End

FIGURE 14

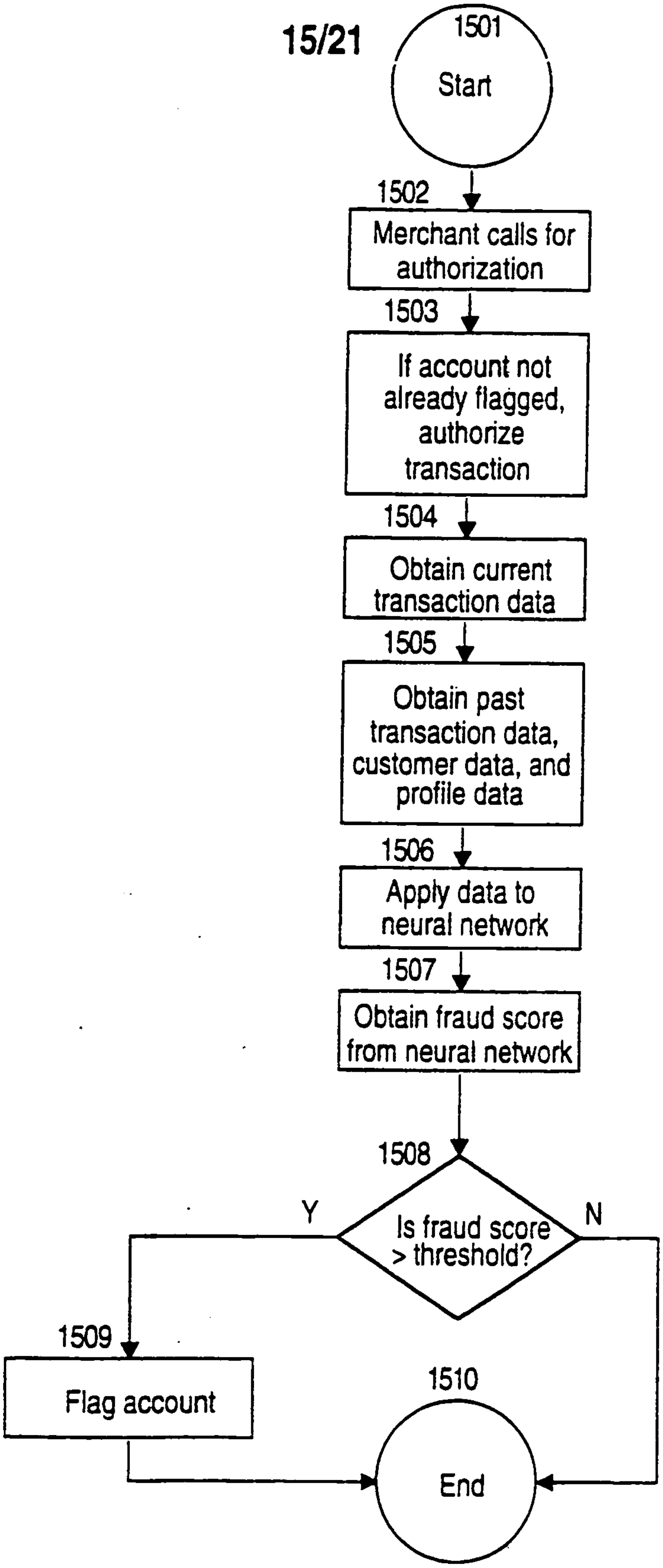

1501
Start
1502
Merchant calls for authorization
1503
If account not already flagged, authorize transaction
1504
Obtain current transaction data
1505
Obtain past transaction data, customer data, and profile data
1506
Apply data to neural network
1507
Obtain fraud score from neural network
1508
Is fraud score > threshold?
Y
N
1509
Flag account
1510
End

FIGURE 15

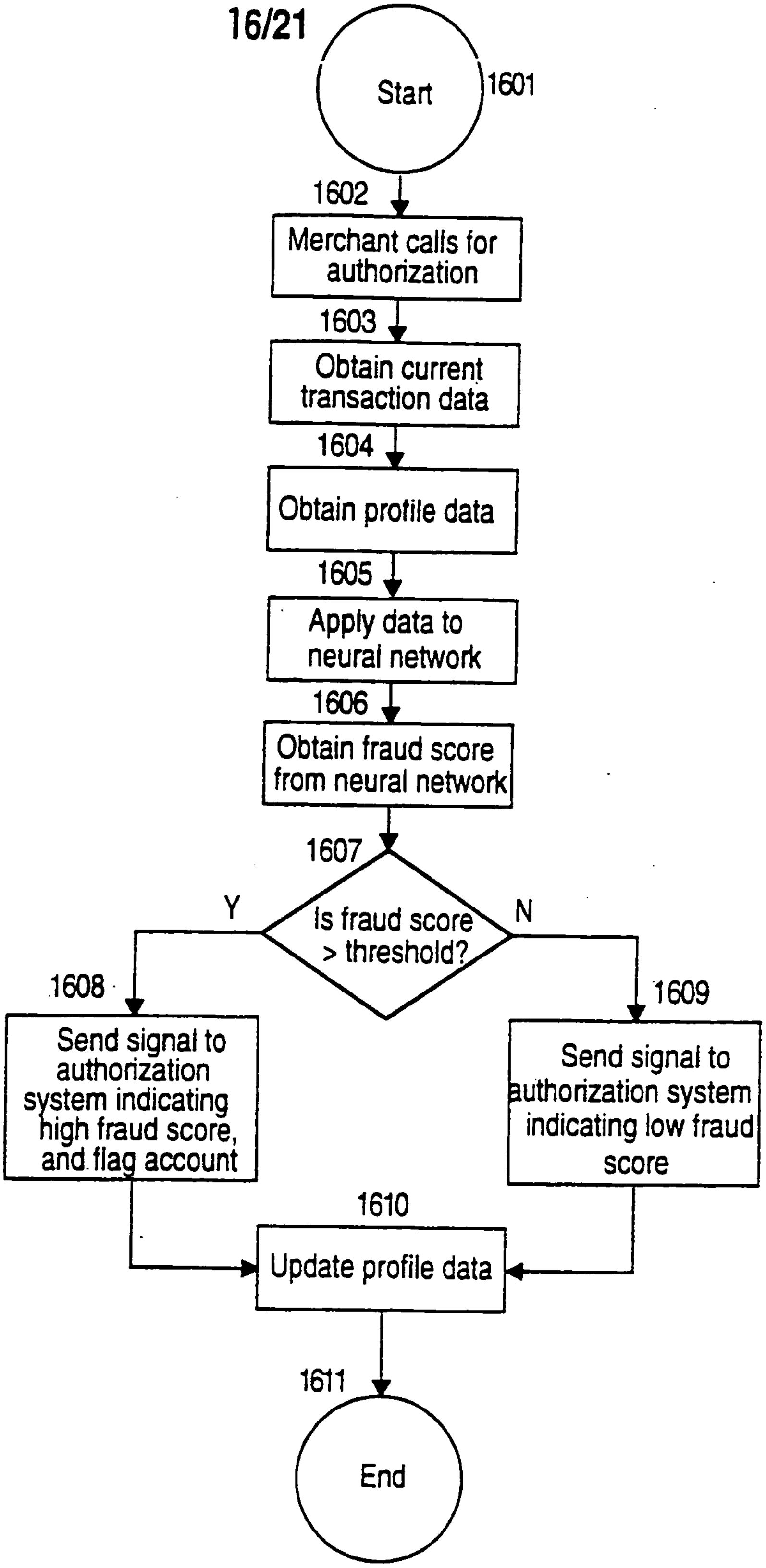

Start
1601
1602
Merchant calls for authorization
1603
Obtain current transaction data
1604
Obtain profile data
1605
Apply data to neural network
1606
Obtain fraud score from neural network
1607
Is fraud score > threshold?
Y
N
1608
Send signal to authorization system indicating high fraud score, and flag account
1609
Send signal to authorization system indicating low fraud score
1610
Update profile data
1611
End

FIGURE 16

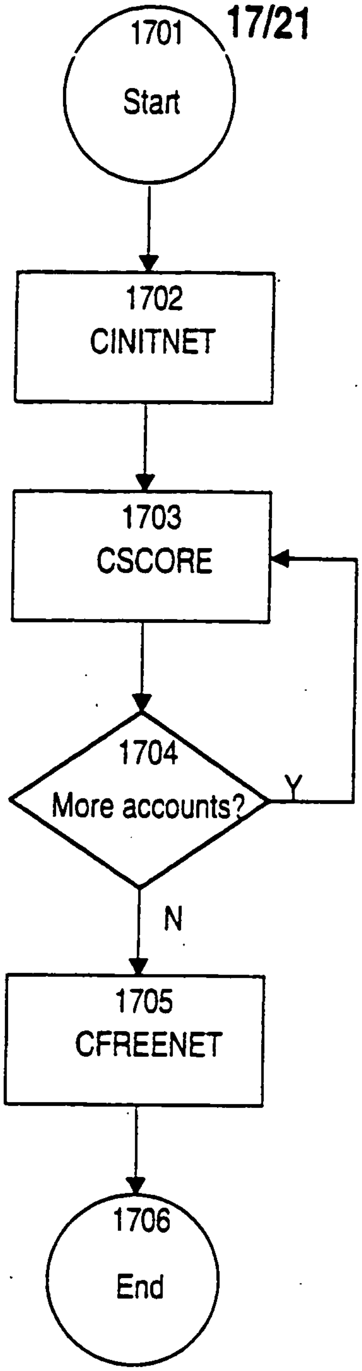

1701
Start
1702
CINITNET
1703
CSCORE
1704
More accounts?
Y
N
1705
CFREENET
1706
End

FIGURE 17

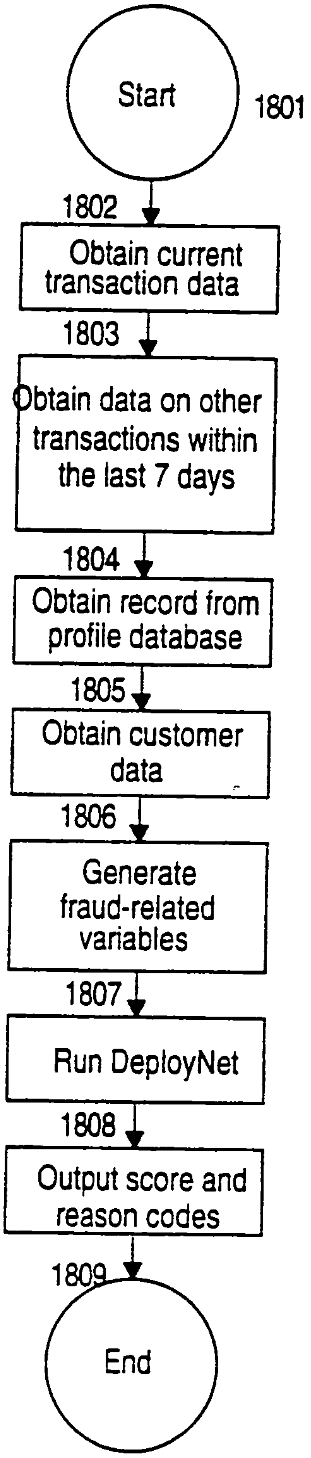

Start
1801
1802
Obtain current transaction data
1803
Obtain data on other transactions within the last 7 days
1804
Obtain record from profile database
1805
Obtain customer data
1806
Generate fraud-related variables
1807
Run DeployNet
1808
Output score and reason codes
1809
End

FIGURE 18

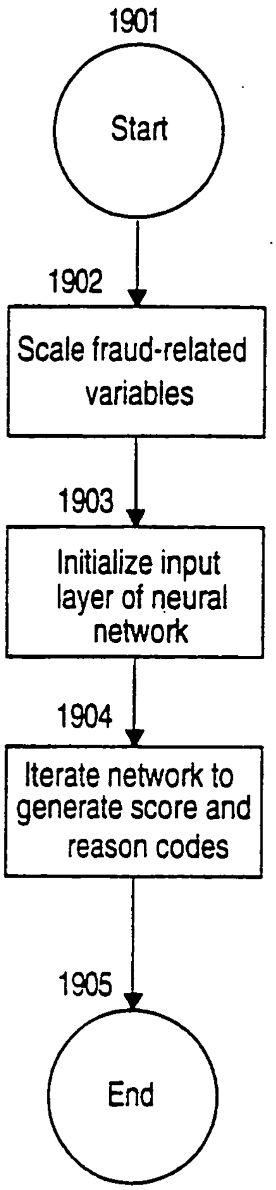

1901
Start
1902
Scale fraud-related variables
1903
Initialize input layer of neural network
1904
Iterate network to generate score and reason codes
1905
End

FIGURE 19

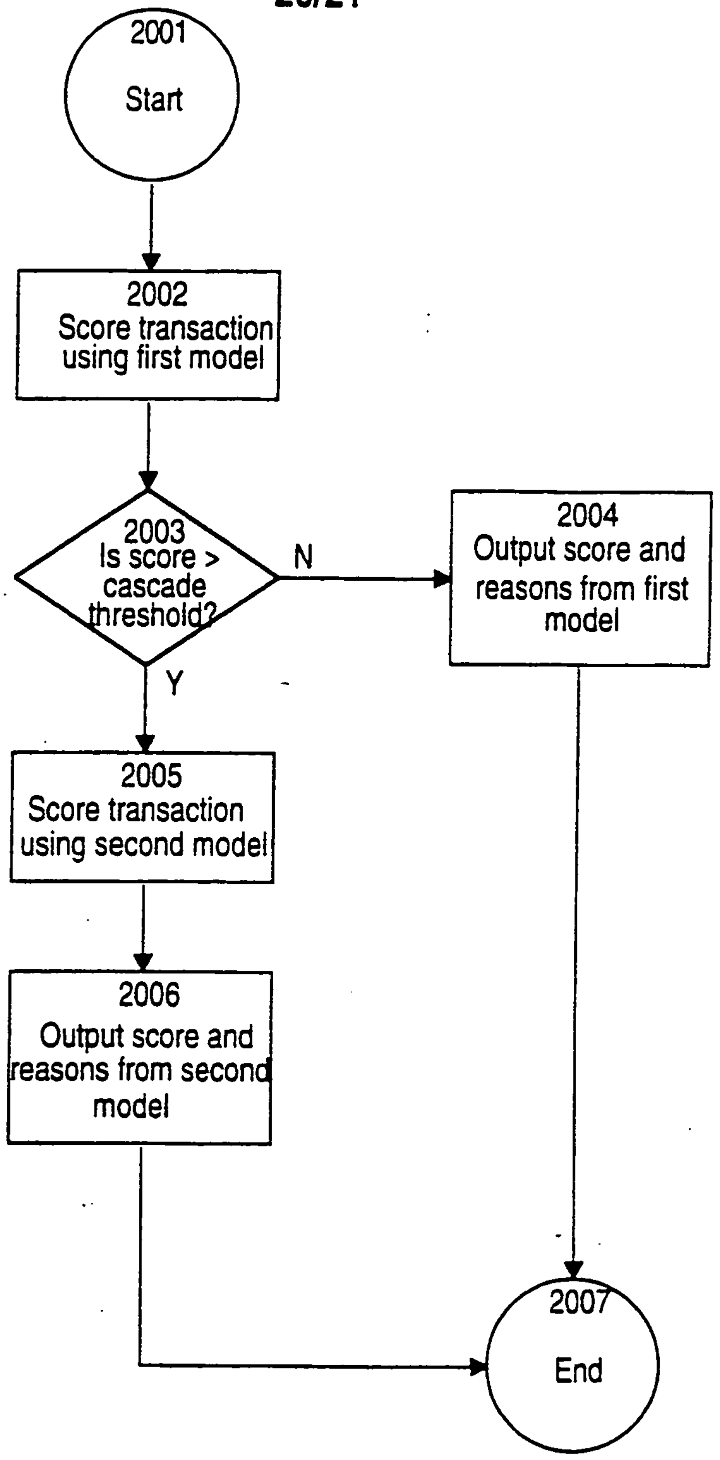
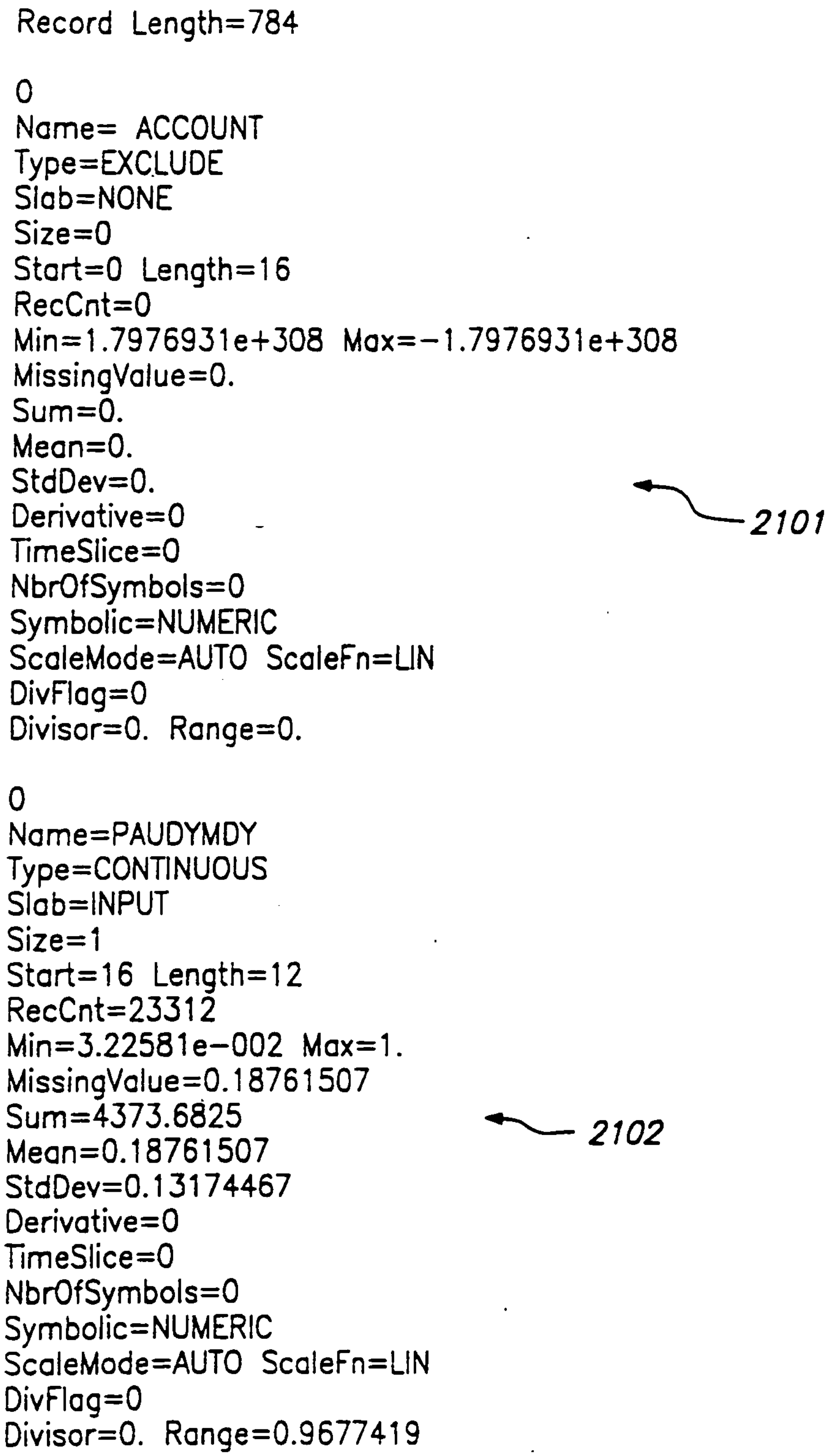

2001
Start
2002
Score transaction using first model
2003
Is score > cascade threshold?
N
Y
2004
Output score and reasons from first model
2005
Score transaction using second model
2006
Output score and reasons from second model
2007
End

*FIGURE 20*



```
Record Length=784

0
Name= ACCOUNT
Type=EXCLUDE
Slab=NONE
Size=0
Start=0 Length=16
RecCnt=0
Min=1.7976931e+308 Max=-1.7976931e+308
MissingValue=0.
Sum=0.
Mean=0.
StdDev=0.
Derivative=0
TimeSlice=0
NbrOfSymbols=0
Symbolic=NUMERIC
ScaleMode=AUTO ScaleFn=LIN
DivFlag=0
Divisor=0. Range=0.

0
Name=PAUDYMDY
Type=CONTINUOUS
Slab=INPUT
Size=1
Start=16 Length=12
RecCnt=23312
Min=3.22581e-002 Max=1.
MissingValue=0.18761507
Sum=4373.6825
Mean=0.18761507
StdDev=0.13174467
Derivative=0
TimeSlice=0
NbrOfSymbols=0
Symbolic=NUMERIC
ScaleMode=AUTO ScaleFn=LIN
DivFlag=0
Divisor=0. Range=0.9677419
```

FIGURE 21

# INTERNATIONAL SEARCH REPORT

Interr ial Application No
PCT/US 93/08400

A. CLASSIFICATION OF SUBJECT MATTER
IPC 5 G07F7/10 G07F7/08 G06F15/21

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 5 G07F G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category * | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| X | EP,A,0 418 144 (MICHAUD) 20 March 1991 | 1,2,5-8,<br>11-17,<br>22,23,<br>28-30 |
| A | see abstract<br>see page 5, line 40 - page 7, line 43<br>see page 9, line 1 - line 22; claim 1<br>--- | 4,18-20 |
| A | EP,A,0 421 808 (MANSVELT) 10 April 1991<br>see abstract<br>----- | 1 |

□ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

| | |
|---|---|
| Date of the actual completion of the international search<br>27 December 1993 | Date of mailing of the international search report<br>12. 01. 94 |
| Name and mailing address of the ISA<br>European Patent Office, P.B. 5818 Patentlaan 2<br>NL - 2280 HV Rijswijk<br>Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,<br>Fax: (+31-70) 340-3016 | Authorized officer<br>Taccoen, J-F |